Semantic Web 0 (2024) 1–0
IOS Press

Ontological analysis of malfunctions: some formal considerations

Francesco Compagno [a,b,*] and Stefano Borgo [a]
[a] *Laboratory for Applied Ontology (LOA), Institute for Cognitive Sciences and Technologies (ISTC), Trento, Italy*
[b] *Industrial Innovation, Università degli studi di Trento, Italy*
E-mail: francesco.compagno@unitn.it
E-mail: stefano.borgo@cnr.it

Abstract.

An engineering device or system is designed to retain its functioning conditions for the time it is needed. Unfortunately, failures happen and mark the state in which a device stops functioning to a satisfactory level and, thus, is considered malfunctioning. The study of malfunction and its prevention is a central topic in engineering and its analysis poses challenges relevant to applied ontology as well.

In this work, we focus on the intersection between the conceptual and ontological understanding of causation and malfunction by studying existing works in these areas. Aiming to compare these approaches from a unified viewpoint, we introduce a causation-based analysis method to take on terminological and conceptual challenges. The goal is to develop a formal taxonomy of malfunctions which can also be used to compare the other approaches in the literature. Our work pays attention to real case scenarios paving the way to the ontological understanding and modeling of failures in practice. Furthermore, it helps exploiting the potentialities of failure analysis and, in particular, of root cause analysis.

Keywords: Ontology, Malfunction, Failure, Failure mode, Causation

1. Introduction

Malfunctioning is a crucial topic in engineering. Failure of engineered systems can carry consequences ranging from a little time lost during the assembly of some product to tragic results including extreme cases like the meltdown of nuclear reactors in Chernobyl, in 1986, and in Fukushima, in 2011, and the release of toxic gas in Bhopal in 1984.

In all these cases, it is desirable to record information about and to analyze the occurrence of the failure and to understand the cause(s) behind it. In this way, further occurrences of the failure can be prevented, or at least made easier to treat, and, if legal matters are concerned, a solid technical understanding can facilitate reaching a settlement regarding blame assignment [1].

The motivation for this paper also arises from the recognition that failure analysis in engineering is often hindered by terminological and conceptual confusion within the domain. Different stakeholders, even though they have expertise in similar or the same areas, might use different terms to describe the same failure phenomenon, leading to miscommunication and reduced efficiency in identifying and addressing the root causes. By conducting an ontological analysis and providing clear definitions for key domain concepts, this research aims to establish a common

*Corresponding author. E-mail: francesco.compagno@unitn.it. Postal address: Via alla Cascata, 56, 38123 Trento TN, Italia.

understanding and language for failure analysis. This standardized approach could bridge the gap between various engineering disciplines, facilitating interdisciplinary collaboration and knowledge-sharing. Moreover, the paper acknowledges the importance of causation in failure analysis, as identifying the precise chain of events that led to a failure is pivotal to identifying the chain of 'responsibility' for the failure, as well as in developing effective preventive measures. By offering a solid technical foundation for failure analysis, this paper seeks to contribute to more robust engineering practices, ensuring safer and more reliable systems in the future.

The paper is organized as follows: the next section presents a literature review where we present our concerns, and discuss and compare some classical works. In section 3 we present our formal proposal for the interpretation of some domain concepts within a single ontology called MALFO (MALFunction Ontology). Afterwards, we will compare our work with relevant existing approaches (Section 4), before concluding the paper.

2. Literature review

2.1. Failure, fault, and malfunction

Failure, malfunction, and fault are some of the terms typically used to indicate that something in an engineered system is not as it should be. In practice, such terms are used with a range of meanings: for example, the term 'failure' can be used to indicate the "condition in which an engineering member exhibits plastic deformation" [1, p.16] or "loss of ability to perform as required "[2, 192-03-01]. In fact, a vast number of different definitions of failure are present in the literature. Del Frate [3] has carried out a detailed analysis of the term failure, classifying numerous definitions into three general categories [4]. Briefly, Del Frate states that there are essentially three concepts of failure (and three corresponding concepts of fault):

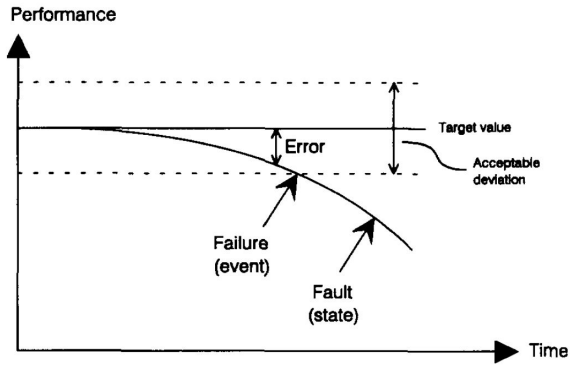
- **Function-based failure**: the termination of the ability of an item to perform a required function [*due to an internal state*]. (The definition is found in the IEC vocabulary [2], the phrase in italics is added by us since it is an important part of the intended meaning).
- **Specification-based failure**: the termination of the ability of an item to perform as specified provided it has been operated under the stated operational environment for which it is designed.
- **Material-based failure**: any permanent change in the values of geometrical or physicochemical properties of the materials of an item which (i) renders the item unable to perform as specified or (ii) increases substantially the likelihood that the item will become unable to perform as specified.

Del Frate also suggests a further definition: "Failure is the inability of an engineering process, product, service or system to meet the design team's goals for which it has been developed" [3], which is argued has good properties, especially in the context of interdisciplinary use. Notice that the difference between the second and the first definition is duplex: the specific-based definition asks for more constraints than the function-based, in particular, malfunctions due to improper use of equipment by the operators are not considered specification-based failures. Moreover, an item could perform a function even when violating design specifications. E.g., Del Frate [4] considers the example of a trailer barrel used to haul chemicals. If the barrel is cracked then is not up to specifications, but if the crack is small and located to the uppermost part of the barrel, the barrel may still carry out its function.

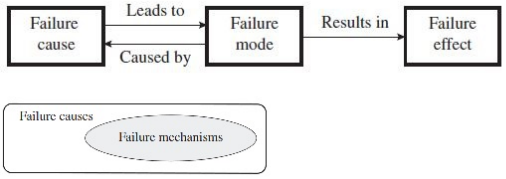
Another frequent distinction in the literature is the one between failure and fault: usually, but not always, a failure is said to be an event, while a fault is said to be a state in which a failure results. Sources that comply with this use are Del Frate [3], the IEC vocabulary [2], and Rausand and Øien [5] (see Figure 1a).

However, some authors propose an opposite interpretation: failures are the results of faults, through the proxy of an error, defined as the deviation between expected and actual states of a system [7]. Such an opposite interpretation need not be irreconcilable (cf. the discussion on Avizienis' work in Section 4), but it is bound to generate confusion. Another different use of the term is, e.g., the one of Vesely [8]: in this case, the term failure has a meaning similar to function-based failure, while the term fault is a hypernym that includes also the possibility that the inability to perform is not due to an internal state but is caused by the failure of a separated component.

A term strongly related to 'fault' and 'failure' is 'down state' [9], or 'failure state' for some authors [10]. The term is defined by the standard EN 13306 as the "state of an item being unable to perform a required function due



(a) A diagram taken from [5] which illustrates the difference between fault and failure. The term 'error' is also present, with the same meaning as in the IEC vocabulary: "discrepancy between a computed, observed or measured value or condition, and the true, specified or theoretically correct value or condition"



(b) Two pictures, taken from [6], which show a possible understanding of the causal relations among failure mechanisms, modes, and effects.

Fig. 1.

to preventive maintenance or a fault" [9]. Given the above discussion, the standard's definition is similar to the common idea of fault: the only difference is that it distinguishes between the fault and the state determined by the fault (though what a fault is, if not a state, is left unspecified).

Finally, the term 'malfunction' is often used as a synonym for failure or fault in engineering use: for example, the IEC vocabulary defines it as "situation for which the electrical equipment does not perform the intended function due to a variety of reasons [...]", which is synonymous with 'being at fault', using IEC's definitions.

Coming to philosophical interpretations of malfunctions, they have been the subject of several analyses: Baker [11] defines malfunctions similarly to the function-based failure concept of Del Frate and asserts that malfunctions, despite being defined by the negation of functionality performance, have the same ontological dignity of artifacts and functions: "There are no technical artefacts without functions; there are no functions without the possibility of malfunction". Jansen [12] highlights the difference between malfunction and malfunctioning, which is analogous to the one between function (a property) and functioning (an occurrence). Then he argues that functions are not dispositions, nor depend on them: this is entailed by this definition of malfunctions, which consist in the presence of a function without "the matching disposition". A somewhat similar take is found in Mizoguchi et al. [13], where it is argued that malfunction can be explained only if functions do not inhere in objects (in [13] functions are understood as "roles played by behaviors in a function context") and in [14], where malfunctioning happens when an artifact does not possess all the capacities (abilities) required from it. Such an approach is opposed to Spear's [15], which makes use of the BFO upper-ontology and states that functions inhere into objects and (complete) malfunctions entail the end of the object life.

2.2. *Failure mode, failure mechanism, and failure effect*

Other common terms in the context of malfunctioning are (failure) mode, effect, and mechanism. These terms are especially important since they have been used in the context of FMEA (Failure Mode and Effect Analysis) from its inception [16]. In particular, failure mode is defined as "[t]he manner by which a failure is observed. Generally describes the way the failure occurs and its impact on equipment operation." [16]. This is significantly different from the more laconic definition given by the IEC vocabulary: "manner in which failure occurs", and even more different than the definition given, e.g., in [17]: "The consequence of the mechanism through which the failure occurs, i.e., short, open, fracture, excessive wear". In practice, exhaustive examples of failure modes can be found in the many tables compiled by reliability engineers, such as [17–20] (e.g. a few terms from [18] are, for an actuator, 'leaking',

'seized', 'worn', 'contaminated', etc.). Analyzing these examples, one can arrive to see that within failure modes are collected quite different things. For example, failures to carry out the required function ('fail to open'), or conditions that are undesired since they may cause failures in the future ('worn', 'vibration').

We conclude by highlighting that there are, at least, two recurrent ways in which failure modes are intended: they may be specific parts of a causal chain (cf. [17, p. 3-6], for which they are 'consequences', or Figure 1b). Such a causal chain is usually composed of a failure mechanism, described as a process that leads to failure [2, 192-03-12], which causes the failure mode, which, in turn, causes a 'failure effect'. The other view describes failure modes as the symptoms immediately perceivable of a malfunction (e.g., "a failure mode is a manifestation of the failure as seen from the outside" [5, p.77], "[t]he effect by which a failure is observed on the failed unit" [20, p. 41]).

2.3. Cause, root cause, immediate cause, and other causes

Causality is critical to the topic of malfunctioning, especially from an application viewpoint. Indeed, to assign blame for a failure (instance), or to associate a failure (type) to a cause (type), or to understand how to prevent the same (type of) failure in the future, one must know the exact cause(s) of the failure in the given case (assuming this can be established). Therefore, various terminologies addressing causes and effects have been used in the literature.

A most common term is simply 'failure cause', specifically, the "set of circumstances that leads to failure" [2], but it is used with many different meanings. Moreover, specialized terms are employed to distinguish between types of causes. For example, 'proximate' or 'direct' [6] or 'immediate' [21] causes are those causes that are immediately apparent and temporally and spatially adjacent to the failure that is being considered. Such terms are sometimes contrasted with 'basic' [21] or 'root' [6] causes, which, on the other hand, may have happened some time before the examined failure, and are considered to grant a more complete answer to the failure analysis .

Root causes form arguably the most important type of causes, since one is often interested in finding the most proper cause of a failure. Many definitions in the literature focus on this type. Del Frate [22] surveyed such definitions, and found two main ways of thinking about root causes: a forward-looking view, which emphasizes the effects that the correction of the root cause will have on the system, and a backward-looking view, which emphasizes the important role of the root cause in bringing about the failure. One critical goal of the forward-looking view is separating the search for a correction from a search for blame assignment; while the backward-looking view focuses on the goal of identifying the right moment when to accept a cause as the root cause, in place of keeping searching for earlier causes. In the end, Del Frate attempted a conciliatory definition, stating that the root cause is "that element of the factors and causes that is, if corrected, the most likely to prevent failure phenomena similar to the one under investigation from happening again".

Turning back to failure mechanisms, Rausand [6] defines them as processes that lead to a failure. In this view, failure modes are intermediate elements of causal chains started by failure mechanisms and producing failure effects, see Figure 1b. One then deduces that failure mechanisms are a subset of failure causes.

In the following sections, we will carry out an ontological analysis of malfunctions. In particular, we will propose a clear and unifying conceptual model for some of the presented terms, as well as discuss the topic of causation.

3. Ontological analysis of malfunctions

In the rest of the paper, we use the term 'failure-related happening' as a catch-all term for any individual (or token)[1] occurrence considered unwanted by domain experts. We also take 'unwanted' and 'abnormal' as synonymous primitive terms to mean that there is a mismatch between what is occurring and the ideal mental model of a domain expert.

Given the discussion of failure-related happenings in Section 2, we claim that there are at least four key aspects that must be highlighted to carry out a successful ontological analysis of these concepts:

[1] So e.g. 'John was diagnosed of cancer yesterday' and not the activity(-type) of diagnosing in general. Our ontology will be about token-level occurrences as discussed in sect. 3.1.

– **Ontological categorisation**: failure-related happenings cover the usual ontological categories of occurrences, namely, Event, Process, and State.
– **Causal relations**: failure-related happenings are defined by the role they take in a causal chain.
– **Granularity**: Failure-related happenings can be directly decomposed (as in [23]) or associated with a decomposition of an engineering system, so that finding a suitable granularity level for failure analysis or determining how to transition from one granularity level to another is critical.
– **'Internality'**: In addition to granularity, it is important to establish whether a failure-related happening is internal to a system or component, or not. This distinction is often essential, as evident in the definition of fault in the IEC vocabulary [2, 192-04-01] (see sect. 2.1), or in the differentiation between internal and external causes (faults) discussed, e.g., in [7, 24].

There are other relevant aspects of failure-related happenings but we limit our analysis to those listed above. In particular, there exist in the literature two recurrent aspects that we call 'culpability' (that is, distinguishing failures depending on who is responsible for them) and 'observability' (that is, the property of a happening of being easily observable by the technicians working with an engineering system). Those two aspects are surely important, but we exclude them from the current analysis: regarding culpability, causal analysis is a solid basis and necessary precondition to establishing culpability, while the opposite is not true. Regarding observability, it is an epistemic aspect, not ontological, so it falls outside the focus of this paper.

3.1. Brief formalization of general causation

Our work builds on the ontology of causation proposed by Toyoshima, Mizoguchi, and Ikeda in [25]. Therefore, we briefly recall the main predicates of the causation ontology, while the complete formalization is detailed in the original paper. A serialization of our work in CLIF [26] and its approximation in OWL, both including (our version of) the causation ontology, are available on GitHub[2]. We anticipate that our (full) theory is consistent (this was verified with the Vampire theorem prover[3]).

It is important to note that the causation ontology is about (causation among) token-level occurrences, not type-level occurrences (occurrents are things that take place in time and are not necessarily wholly present at any time they are present). This means that a statement like 'smoking causes cancer' is not a target of the ontology. While expressions like, e.g., 'John has cancer because he smoked for 20 years' can be modeled. Therefore, the same holds true for our work. The core of the causation ontology is the causation relation, modeled as an irreflexive and asymmetric binary predicate that holds between occurrents. As mentioned above, such a causation relation holds between occurrence-tokens, not occurrence-types.

Typical types of occurrents are states, processes, or events, which we indicate with the unary predicates `State`, `Process`, and `Event`, respectively. In engineering, states are often described by specifying some characteristics (form, value of a process variable, ...) of the entities participating in the state (e.g. 'the shaft is broken', 'high oven temperature', or 'the oven temperature is increasing'); while a typical example of a process is a sequence of states (or, equivalently, of state transitions) related to some physical phenomena. For instance, corrosion is a process made of a sequence of many single chemical reactions that change the arrangement of the relevant atoms. Finally, typical examples of events are those occurrences that, being very brief, are considered instantaneous, such as impacts or ruptures.

In our ontology, there are no characterizations of occurrents and their subclasses (except for the fact that states, processes, and events are disjointed subcategories of occurrents) since we aim to (i) preserve compatibility of this work to multiple possible upper-level ontologies; and (ii) avoid the genuine ontological problem of characterizing occurrents (which goes beyond the scope of this paper[4]). Additionally, the causation ontology of [25] distinguishes between actual and non-actual events, while we, for the sake of simplicity, suppress this distinction.

[2] https://github.com/kataph/MALFO. The repository contains CLIF, TPTP, Prover9, and OWL serializations of our full ontology.
[3] https://github.com/vprover/vampire.
[4] The interested reader can consult appropriate reviews, e.g. [27], to see the main ways in which occurrents are classified in various ontological approaches.

The causation ontology of [25] distinguishes causation along two axes: causation can be either direct or indirect, and positive or negative. Direct and positive causation is called `achieves` and requires tight coupling between the arguments, in particular the juxtaposition or overlapping of their time extension. Direct and negative causation is called `prevents`. Indirect and positive causation is called `allows`, while indirect negative causation `disallows`. The main claim is that any sort of causation can be reduced to these four relations.

Among these four, achieve is the only primitive relation, the others are defined from it (and other auxiliary notions). For instance, `allows` is defined as achieving a state that is incompatible with the caused occurrence, where incompatible-with is a primitive of the theory that roughly indicates the physical impossibility of coexistence

Coming to indirect causation, it is founded on the concept of 'state-mediation', meaning that it postulates that there can be states that are necessary or probable (pre-)condition for the happening of an occurrence (we write the relation between any of these states and the occurrence as `facilPreconditionFor`). Similarly, a primitive is used to state that a state is a (pre-)condition preventing or obstacling the happening of an occurrence (`prevPreconditionFor`). `allows` is then defined as either (i) achieving or maintaining a facilitative precondition, or (ii) preventing a preventive precondition (and assuming that, at the same time, no additional preventive precondition was achieved). Finally, `disallows` is the negative counterpart of `allows`, with 'facilitative' and 'preventive' exchanged.

Additionally, not all types of occurrents can be related by the `achieves` relation and the other causal relations: for example, processes cannot achieve states, and events are never achieved. The reasoning for this, as well as for other finer points, such as the contextuality of the causal relations, is detailed case by case in [25], and we do not report it here. We recall that the full axiomatisation is available through a common logic serialization[5].

Extensions of the causation ontology In the original causation ontology, one cannot say that an occurrence is internal to another occurrence. Here "internal" means that the spatial location of the first occurrence is included in the spatial location of another occurrent or of an object. We need this, hence, we add to the causation ontology a further primitive holding between occurrences:

pr1 `internalTo`(x,y)
"*x* is internal to *y*." For example, consider the case of the right headlight of a car that is not working because the filaments of its light bulb are burned. The occurrence 'the filaments are burned' (in other words, 'the filaments are disconnected') is internal to the occurrence 'the right headlight is not working'.

The `internalTo` relation might be defined using, say, a formal ontology supplying both a mereology of objects and an object-occurrence participation relation, but this depends on the foundational or reference ontology one adopts. This issue is outside the scope of the present paper. Additionally, this predicate could be more complex than at first glance: consider the case that, in a chemical plant, a part of a device breaks off and flows downstream, until it arrives in the chamber of a valve, making the valve stuck and causing malfunctioning. Then, considering the occurrence "there is a foreign object in the valve" as internal to the valve (after all the corresponding spatial locations are subset one of the other), together with a definition of fault (e.g., definition (df8) given later), would entail that the valve itself is at fault, while the standard interpretation would say that it is the broken component that is at fault. We contend that in this case one should *not* say that `internalTo` holds because the foreign object is not part of the valve (contrarily to the example of the burned light bulb filaments). This should suffice to make our point. We do not deepen this discussion further since it is outside the scope of this paper.

3.2. Taxonomy of failure-related happenings

At the start of this section, three key aspects of failure-related happenings were identified. We exploit those to build a taxonomy, which is the core of our ontology MALFO (MALFunction Ontology), to classify token-level occurrents arguably related to failures. To collect all such token-occurrences, the top of the taxonomy (excluding 'Occurrent') corresponds to the term 'failure-related happenings', which we use as a catch-all term[6]. The taxonomy

[5] https://github.com/kataph/MALFO.

[6] Ontologically speaking, the class 'Failure-related Happening' is introduced as the subclass of occurrents that are relevant for the work in this paper. The introduction of this class does not imply that these occurrents have a special ontological status.

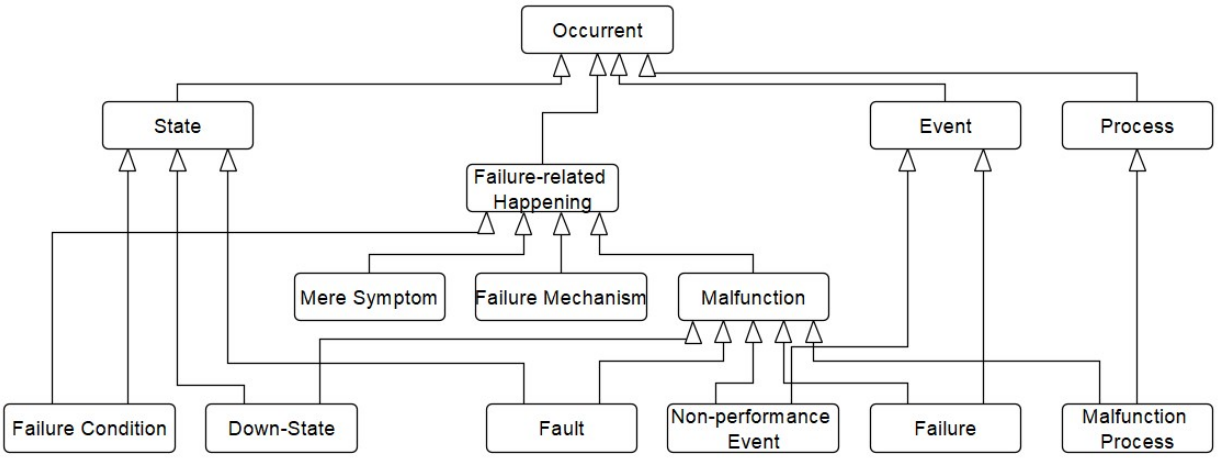


Fig. 2. A pictorial representation of the taxonomy introduced in this paper.

is represented in Figure 2, while formal definitions for the elements of the taxonomy are listed and discussed below.

We start with a predicate, `FunctionCompatible`, which we use to identify those failure-related individual happenings that do not immediately interfere with the nominal functionality of artifacts. We define such a predicate exploiting previous work on functionality, based on the upper-ontology DOLCE [28]. in particular, we use the concept of "systemic function" that was therein defined. To simplify the reading experience, we report here the original definition and its elucidation verbatim (df1)[7]. The definition contains some predicates that we have not discussed, but are described in [28]. More precisely, `R` is DOLCE's category for roles, `CF` ('classified by') is a temporalized relation holding between a role and its player, `participatesAsDoer` is a specialization of the `participatesIn` relation (DOLCE's relation to indicate the participation of objects in occurrences) which denotes the role of an agent or agent-like 'doer' w.r.to the participated occurrence. Finally, two predicates are used to indicate systems and their goals, while `causalContr` is a generic causal relation between occurrents, which we can better refine in light of the causation ontology employed in this work (see (pr2) and its elucidation). Using (df1) we can then define `FunctionCompatible` (df2). The predicate `FunctionCompatible` could be introduced as a primitive without committing to the work in [28], the remaining definitions do not formally change.

df1* $\texttt{functionOf}_{\texttt{Sys}}(x,y) \coloneqq \texttt{R}(x) \wedge \exists z,g,b,t(\texttt{System}(z) \wedge \texttt{goalOf}(g,z) \wedge \texttt{CF}(b,x,t) \wedge \texttt{P}(y,z,t) \wedge \forall b',t'(\texttt{CF}(b',x,t') \implies (\texttt{Behaviour}(b') \wedge \texttt{participatesAsDoer}(y,b',t') \wedge \texttt{P}(y,z,t') \wedge \texttt{causalContr}(b',g))))$

"x is a systemic function of y if and only if x is a role and there exist a system z and a goal g for z such that x classifies only behaviours which have y as doer and part of z, and that causally contribute to achieve g."

df2 $\texttt{FunctionCompatible}(x) \coloneqq \texttt{Occurrent}(x) \wedge \forall o,f,b,t((\texttt{participatesIn}(o,x,t) \wedge \texttt{functionOf}_{\texttt{Sys}}(f,o) \wedge \texttt{CF}(b,f,t)) \implies \neg\texttt{incompatibleWith}(x,b))$

"x can coexist with normal functionality (in short, is function-compatible) if and only if it is an occurrent and, for each object (o) participating in x at some time (t), if that participant has a systemic function (f) at that time realized through a behaviour (b), then, the behaviour is not incompatible with x." Notice that in the previous definition b must be the behaviour of o (the behaviour o participates-in-as-doer) due to Definition (df1).

With `FunctionCompatible` we mean synchronous logical-compatibility of an occurrence with the execution of the functions of the components mentioned in the occurrence. For example, the presence of a light beam in front of a car's headlights is compatible with those headlights' (and the car's) function. We mean synchronous in the sense that the presence of a (small) crack can coexist with the normal functionality of an

[7] In the original paper the definition was labeled **d8**. Note that (df1) has a star: it means that it is not mandatorily part of our malfunction ontology.

axis of transmitting torque, even if that crack could cause a fracture in the future, while the fracture itself would then not be compatible with the axis functionality[8]. Another example: if the function of an electric wire is to carry electricity, and a logically-necessary requirement for the wire of doing that is being physically connected, then 'the wire being cut in two' is *not* `FunctionCompatible` with the wire's function. Notice that, in reality, the compatibility could be complete or partial, but, for the sake of simplicity we do not discuss this, though it is something compatible with our theory. In addition, `FunctionCompatible` is intended to be just as a shortcut for its definient, and not as a particularly relevant concept: we did not include it into Figure 2.

df3 $\texttt{posCauseOf}(x,y) \coloneqq \texttt{achieves}(x,y) \vee \texttt{allows}(x,y) \vee \texttt{facilPreconditionFor}(x,y)$
"x positively-causes y if and only if x achieves or allows y or is a facilitative precondition for y."

pr2 $\texttt{posCauseOf}(x,y)^{\dagger}$
"x eventually positively-causes y."
With this primitive, which will be used in the next axiom, we indicate the transitive closure of the `posCauseOf` relation (transitive closure cannot be expressed in first-order logic, so we introduce it as a primitive).

df4 $\texttt{FailureMechanism}(x) \coloneqq ((\texttt{Process}(x) \vee \texttt{Event}(x)) \wedge \texttt{FunctionCompatible}(x) \wedge \exists y(\texttt{Failure}(y) \wedge \texttt{posCauseOf}(x,y)^{\dagger}))$
"x is a failure mechanism means that x can coexist with normal functionality, is either a process or an event, and is the positive cause, eventually, of some failure." `FailureMechanism` tries to convey the meaning behind common occurrence-types such as corrosion, wear, cracking, etc., which typically are not considered failures, but do cause failures in some sense.

df5 $\texttt{FailureCondition}(x) \coloneqq (\texttt{State}(x) \wedge \texttt{FunctionCompatible}(x) \wedge \exists y(\texttt{Failure}(y) \wedge (\texttt{posCauseOf}(x,y)^{\dagger} \vee \exists z(\texttt{prevPreconditionFor}(x,z) \wedge \texttt{disallows}(z,y)))))$
"x is a failure condition means that x can coexist with normal functionality, is a state, and is either the facilitative precondition for a failure or the positive cause, eventually, of that failure, or it is the preventive precondition for some occurrence that disallows the failure." Failure conditions have the same causal role as failure mechanisms, but are states in place of processes (e.g. 'being corroded' in place of 'corrosion'). We also add explicitly the case of negative causation, that is, e.g. the case that the state of being corroded inhibits a safety function and such inhibition then causes a failure.

df6 $\texttt{MereSymptom}(x) \coloneqq (\texttt{FunctionCompatible}(x) \wedge \neg\exists y(\texttt{Failure}(y) \wedge \texttt{causeOf}(x,y)) \wedge \exists z(\texttt{Fault}(z) \wedge \texttt{causeOf}(z,x)))$
"x is a mere symptom means that x is function-compatible and is caused by a fault, while it has no causal relation to any failure." A mere symptom has no causal effect on a system and is therefore differentiated from other occurrence types. For example, if the right headlight of a car is not lit up when it should, the fact that 'there is no light beam shined by the headlight' is a mere symptom because the absence of light, by itself, will not cause the failure of other components[9]. The negative causation in `causeOf` is considered because the symptom could be the absence of some occurrence that, say, the fault `prevents` (e.g., the missing light in the previous example).

df7 $\texttt{Malfunction}(x) \coloneqq \neg\texttt{FunctionCompatible}(x)$
"x is a malfunction means that x can not coexist with normal functionality."

df8 $\texttt{Fault}(x) \coloneqq (\neg\texttt{FunctionCompatible}(x) \wedge \texttt{State}(x) \wedge \exists z(\texttt{internalTo}(z,x) \wedge \texttt{achieves}(z,x))$
"x is a fault means that x is not function-compatible state and is achieved by some occurrence internal to itself." Notice that we somewhat weakened the classic definition of fault "inability to function due to an internal state", otherwise the z in the definient of (df8) should be classified as a state. We consider that such classification is not necessary, and the z may belogn to other occurrent-types: for example, if 'catastrophic

[8]Note that if we speak both of the fracture and of the occurrents instantiating the axis' functionality, then one or the others are non-actual occurrents, since they cannot occur together, as they are incompatible.

[9]We exclude the case that the absence of light when driving causes a crash, because, say, the car is being repaired by a mechanic.

fracture of the main axis' (z) causes an industrial fan to 'be completely non-operational' (x), the first occurrence (the fracture) is an event internal to the second occurrence (being non operational), a state, which is non function-compatible, and, therefore, a fault by definition (cf. the failure analyzed by [1]). Another example, this time in which the z is a state, is the 'being cut into two parts'-state (z) of an electrical wire[10], which is an internal state that causes the state of 'being unable to transmit electricity' (x), which is then a fault by definition.

df9 $\texttt{DownState}(x) \coloneqq (\neg\texttt{FunctionCompatible}(x) \wedge \texttt{State}(x) \wedge \neg\texttt{Fault}(x))$

"x is a down state means that x is not a function-compatible state and also not a fault." The reason for introducing this term is to distinguish between the abnormal states of a component that require corrective actions from those that do not, since the effective cause of the fault is located elsewhere. For instance, if the right headlight of a car is not lit up when it should be, there are at least two possibilities: either the headlight itself is broken, say the lightbulb is blown, in that case 'the headlight is not lit up' is a fault and the headlight must be repaired; or there is another component that is broken, say the wire linking the headlight to the car's battery is cut. In this latter case, it is the wire that must be replaced, and 'the headlight is not lit up' is a down state.

df10 $\texttt{Failure}(x) \coloneqq (\texttt{Malfunction}(x) \wedge \texttt{Event}(x) \wedge \exists y(\texttt{Fault}(y) \wedge \texttt{achieves}(x, y)))$

"x is a failure means that x is a not-function-compatible event that achieves some fault." With this definition we try to capture the common concept of failure present in the literature: the loss of the ability to perform a function. We also stipulate, by definition, that failures cause faults. For example, if a wire, whose function is to carry electricity, is gnawed by a rat, the wire fails when it breaks and ceases to be able to conduct electricity[11].

df11 $\texttt{NonPerformanceEvent}(x) \coloneqq (\texttt{Malfunction}(x) \wedge \texttt{Event}(x) \wedge \neg\texttt{Failure}(x))$

"x is a non-performance event means that x is a malfunction and an event, but not a failure." An example of non-performance event occurs when attempting to turn on the right headlight of a car if its lightbulb is broken: 'the right headlight did not turn on' is a non-performance event, since it is a malfunction but, by itself, does not cause additional failures.

df12 $\texttt{MalfunctionProcess}(x) \coloneqq (\texttt{Malfunction}(x) \wedge \texttt{Process}(x))$

"x is a malfunction process means that x is a malfunction and a process." We introduce malfunction processes to partition completely the malfunction category. For example in Avizienis' [7] an example of a failure in an integrated circuit is described: a faulty IC is invoked, and this causes, at different time and locations, the presence of wrong values in the calculation process. In our taxonomy, we classify this 'erroneus calculation process' as a malfunction process.

Here we summarize the rationale for the category names. 'Failure' and 'fault' are widespread names and their definitions correspond to typically associated meanings. 'failure mechanism' is another well-known term and its definition attempts to highlight the causal role of failure mechanism w.r.to failures. Since failure *mechanisms* are often associated with processes, we introduced the term 'failure condition', to indicate *states* instead. 'Down state' is another term recognized in the literature [9, 10], and we use it to indicate the inability to carry out functions *not* due to an internal state (as opposed to faults). 'mere symptom' highlights the fact that the underlying occurrence does not cause a failure, but is only a sign of a failure. Finally, 'malfunction' is taken, quite literally, as any occurrence that is incompatible with nominal functionality, and, to ensure that the sub-categories of malfunction form a partition, 'malfunction process' is introduced.

Axioms (df2) to (df12) constitute our formal taxonomy. As mentioned before, this taxonomy is about token-level occurrents somehow related to failure, and its categories are defined (mainly) depending on the causal relations their

[10]Notice that, if we were to introduce in our conceptualization performance parameters that allow us to speak about acceptable/unacceptable functional performance, we could talk about 'partial functional incompatibility': when an occurrence is incompatible with the execution of a function within specified nominal bounds. In that case, we could also talk of 'partial faults', for instance, if the wire is not broken but, say, half of its cross section has been cut away, then the wire has a 'partial' fault. As said before, we are not concerned with these matters in this paper.

[11]Of course, if we consider other functions of the wire, for example the protective function of the isolation, the wire fails when the isolation is pierced and the live copper is exposed. Additionally, if we consider the performance of the wire as indicated also by its cross-sectional area (which is inversely proportional to the electric resistance of the wire) and consider 'partial faults' (see Note 10), then the wire fails before being completely gnawed through.

instances have w.r.to other token-occurrents. Note that we decided not to define failure-mode because we consider the term intrinsically vague. One could use the term failure-*mode* at least with the following meanings:

- failure-mode as a *subtype* of failure-related happenings. So that any term of our taxonomy, or even more precise domain terms such as 'rupture', 'overheating', 'wear', etc., could be considered a failure mode: 'mode', or 'manner', in the sense of 'type'.
- failure-mode as any *attribute* predicated on a failure. E.g. a *frequent* overheating, or a *brittle* bending.
- failure-mode as a specific *role* of an occurrence in a causal chain of failure-related happenings. This is the case hinted by figures 1a and 1b, and entailed (implicitly) in e.g. [29]. Therefore, failure mode would simply be the name of the abnormal occurrence focused on in a given causal chain and granularity level. In particular, failure modes, effects, and causes could identify the same occurrences, though in different contexts. In this case, 'mode' would be contrasted with 'cause' and 'effect', which is not standard use in the English language.

In the oncoming section (Section 4) we will compare our theory with some relevant taxonomies present in the literature.

3.3. Finer causal roles

In the previous section, we have named some concepts (e.g. `Failure`) depending on their being cause of some other occurrence. In all these cases, such concepts do not require (or suggest) comparison with other causes. In this section, instead, we discuss the case of causes that can be compared to other causes in some way.

In Section 1, we briefly mentioned immediate causes, we could define those as the occurrences that are related to an effect through the `causeOf` relation. In this case, the term 'immediate' would be contrasted to causes further positioned within a causal chain, meaning that non-immediate causes would be, at most, the occurrences related to an effect through the transitive closure of the `causeOf` relation. This direction of inquiry, though, is only interesting if it discusses cause-types intermediate between immediate-cause and transitive-cause, as the latter are too weak, and the former too strong a concept.

Root causes are the chief example of such a cause-type. There are some difficulties, though, in defining such a concept. First, we cannot rely on the forward-looking viewpoint of Del Frate, which consists of a mental experiment on the possible effects of amending the candidate root cause, while Toyoshima's causal ontology only talks of actually-happened occurrences. Therefore, we should go in the direction of the backwards-looking viewpoint. This would require us to try to discern the relative contributions of different causes to an effect.

Effective sufficient causes However, this is still something difficult to do, especially from an ontological viewpoint. What does it mean, after all, that some occurrence is the 'true' cause of some effect? Simple examples would seemingly deny the possibility of such attribution. For instance, if a force pushes an object, determining its movement, we would say that the 'real' cause of the movement is the application of the force. But, in reality, the application of force may be necessary (if we specify that with the movement of the object we mean its acceleration), in a physical sense, but it is surely not sufficient: the presence, say, of a wall would prevent the movement of the object, or the static friction with the floor could be excessive. The absence of a wall, in turn, is not sufficient but is clearly necessary, so that it would appear on the same level as the application of force, which is counterintuitive.

The authors of the causation ontology recently proposed[12] a development of the ontology to tackle this issue. In particular, it is proposed that genuine sufficient causation, interpreted as the `achieves` predicate, happens only between an event and a state or between two states. That is, either an event achieves a state (e.g. 'the deer was killed' event achieved 'the dear being dead' state) or a state achieves another (e.g., 'the valve being half-open' achieves 'the flow rate being half of the capacity'). Moreover, all other apparently genuine sufficient-causes/achievements, are in reality particular examples of indirect causations (necessary-causes/'allowments'), as is the case of the application of force for the movement of a body. The idea is that it is not true that the application of force is sufficient to move a body, as explained before, but it does seem so, because it is standard to assume a context in which numerous states are present, constituting a set-up for the movement of the body, in such a way that when the force is applied the body

[12] Riichiro Mizoguchi, personal communication, March 2023.

moves. Precisely, these states negate all possible preventive-preconditions-for the body movement while including all the facilitative-preconditions-for the body movement except for one: the current application of a force to the body. When an event occurs that achieves the last precondition (say, 'I started pushing the body', which achieves the state that 'a force is being applied to the body') then the body starts to move. Therefore, it appears that the event 'the body started to move' (or the state 'the body is moving') was achieved by the event of the application of force, while the latter event only allowed the former event (or state). In this case, the event that apparently achieves causation should be called 'effective sufficient cause', which effectively achieves the effect. Which are the causes that are effective and which are not clearly depends on the context, but the point is that, ontologically, only an accumulation of `allows` (i.e., accumulation of `facilPreconditionFor` and removal of `prevPreconditionFor`) exists in the effective-causation case.

We do not discuss this further (a similar and related discussions can be found in [30, p. 11 and onwards]) and leave the ontological analysis of this topic to future works. For this reason, we also do not supply formal definitions of root-cause (we will, however, define causes intermediate between immediate-causes and transitive-cause: see (df15)-(df17)).

4. Aligning concepts in the literature with our ontology

In this section, we discuss how our ontology can be linked to several failure-related concepts present in the literature. Except for the material in [24], we describe such links only informally since the concepts in the literature are given only informally.

Del Frate and Jansen's definitions Our definition of failure (df10) fits well with the function-based definition of Del Frate. The main difference between ours and Del Frate's definition is that in the former we speak of bringing about the inability to execute a required function, while the latter mentions the termination of the ability to execute a required function. It could be argued that such phrases are equivalent (one is a sort of double negation of the other). In any case, we do not mention 'ability to execute a required function' since we focus on abnormal conditions only.

Coming to specification-based failures, one key reason to introduce such a definition is shifting the culpability of a failure from the manufacturer of some equipment towards the user, in the case that the user did not operate the equipment within its specifications. Since culpability is outside the scope of our ontology, we do not discuss a mapping with this category.

Regarding material-based failures, following the definition itself, we distinguish two cases: (i) if the change of the material properties renders the item unable to perform we have a subcase of function-based failure, which could be associated, in our ontology, to a particular subtype (or a list of subtypes) of the `Failure` class (e.g. 'rupture', 'deformation[-event]', etc.). (ii) The second case happens when the change increases the likelihood of a failure. We cannot faithfully represent this case, since we did not consider stochastic relationships, only causation between individual occurrents. Instead, we can represent in our theory a case where a change in the properties of the materials brings about a failure.

Coming now to Jensen's works, his concept of malfunction corresponds arguably to our `Fault` predicate, while its concept of malfunctioning corresponds roughly to our `NonPerformanceEvent`. The main difference between the respective definitions is the absence of dispositions in our ontology, while Jensen uses them in his definitions.

OREDA failure mode taxonomy The OREDA project handbook [20] describes another subdivision of failure modes, which is a de-facto taxonomy that we pictorially represent in Figure 3. Failure modes are described as *observed effects of failures*, so that 'observability' seems to be key aspect of the top node of the taxonomy. Since, as we have already stated, our ontology does not consider such an aspect, we cannot align with such a definition.

Coming to the taxonomy's leaf nodes, "demanded changed of state is not achieved" is arguably mapped as a subclass of `NonPerformanceEvent` ('fail to start' is an example of occurrence-type common to both classes). Then, "undesired change in manner of operation" is said to be a state but also exemplified with "spurious stop", which we would consider an event. Therefore, with the goal of mapping, we would suggest dividing this case into "alteration in the manner of operation" (e.g. "high output"), to be considered a subcase of malfunction states (i.e., either faults or down states); and a remaining part, say "interruption of operation", containing e.g. "spurious stop",

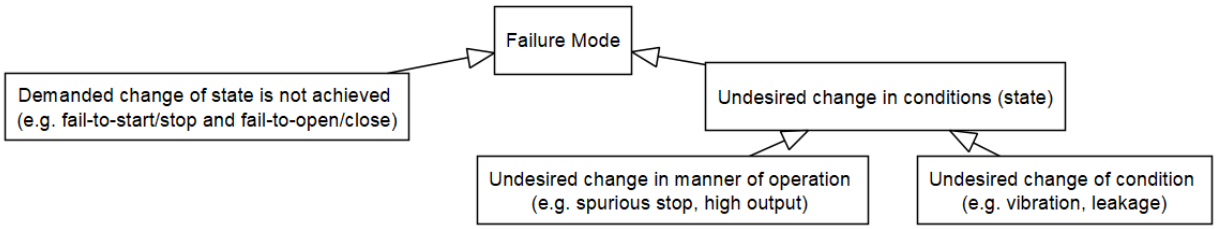


Fig. 3. Taxonomy of failure modes, as described in [20, p. 41]. Note that about the category in the bottom right corner is remarked that "This category does not affect the function immediately, but may do so if not attended to within a reasonable time".

to be mapped into `NonPerformanceEvent`. Finally, "undesired change of condition" should be considered a subclass of occurrences that are states and `FunctionCompatible`. Notice that while we cannot express the fact that instances of this latter category *may* affect the underlying function if not attended to, such a thing could *actually* happen in a model of the theory. Additionally, we could also state that attending to the undesired condition `prevents` a failure from occurring.

Avizienis' fault-error-failure chain As mentioned previously, Avizienis [7] states that faults cause failures (via errors). This may seem incompatible with our definitions, which, following Del Frate, and Rausand and Øien state the opposite: failures cause faults. This is not a contradiction, though, since it is still possible for our ontology that a fault causes a failure and, analogously, Avizienis states that failures may cause faults [7]. That is, Avizienis' characterization of failures and faults is different than ours', but such characterizations may be further specialized in order to be compatible.

Another difference between Avizienis' conceptualization and ours is that he specifies that faults can be dormant or active: an active fault is producing an error, a dormant fault is not. In our ontology, this could be roughly mapped as follows: an active fault is the part of a `Fault` that takes place during (or immediately before) the causation of an error by the fault, while a dormant fault is the remaining part.

The presence of errors is another critical difference. We did not define such a concept in our ontology, partially because Avizienis' definition and examples of errors, as well his statement that "error propagation within a given component [...] is caused by the computation process" [7] suggest that (Avizienis') error is a term specific to the software domain. Nevertheless, we claim that it is possible to translate any Avizienis-style causal chain into a causal chain using the language of our ontology, preserving the most part of the indended meaning: it is sufficient that errors are mapped to some kind of malfunction, depending on the occurrence-type (event, state, or process) the error is recognized as.

Finally, for Avizienis, errors propagate within a component, resulting in other errors, and only when an error reaches the boundary of a system a failure occurs. Given this, we argue that Avizienis' distinction between failures and errors (also) depends on what we called observability in Section 3: the difference between errors and failures is that the latter are observed by some external system. Our theory does not cover observability, thus we cannot explain this difference at this stage.

Kitamura and Mizoguchi ontological analysis of faults The last conceptualization of failure-related happenings we are interested in is the one of Kitamura and Mizoguchi [24], which introduces a rich classification of faults and causes, many of which our ontology cannot represent. Due to the sizeable number of its terms, we discuss only a subset of those concepts that we can align with our ontology.

One key distinction among these concepts is the one between external and internal (with respect to a given system) causes. We can reproduce this distinction by using our `internalTo` predicate (that is, an internal cause of theirs is a cause of ours which is internal to a given system, etc.). Kitamura and Mizoguchi's fault ontology also shares with ours the same distinction between down states and faults, which they call "abnormal states" and "faults", respectively: abnormal states are abnormal "only because of the propagation of abnormal input", while faults are irreversible changes in the (internal) state of a component.

Finally, Kitamura and Mizoguchi propose a crucial distinction among causes of a failure-related happening: they speak of "relative", "absolute", and "ultimate" causes. The distinction between relative causes and absolute causes

is based on the difference between faults and abnormal states: the latter are considered superficial symptoms, which are propagated by a series of "propagation events" started by a fault. The last propagation event is the relative cause of the abnormal condition, while the cause of the fault is the absolute cause of the abnormal condition. A fault may have been caused by a causal chain containing many faults and abnormal states. If that happens, the first cause of a fault which is a cause external to the system under analysis is called the ultimate cause.

Given this, we obtain the following (formal) definitions of the concepts in Kitamura and Mizoguchi's ontology (each labeled with a star $*$) in terms of ours[13]:

df13* $\texttt{AbnormalState}(x)^* \Longleftrightarrow \texttt{DownState}(x)$
"Kitamura and Mizoguchi's abnormal states are equivalent to our down states."

df14* $\texttt{Fault}(x)^* \Longleftrightarrow \texttt{Fault}(x)$
"Kitamura and Mizoguchi's faults are equivalent to our faults."

df15* $\texttt{relativeCause}(x,y)^* \Longleftrightarrow \texttt{posCauseOf}(x,y)$
"Kitamura and Mizoguchi's relative causation is equivalent to the `posCauseOf` relation."

df16* $\texttt{absoluteCause}(x,y)^* \Longleftrightarrow (\exists z(\texttt{Fault}(z) \wedge \texttt{posCauseOf}(x,z)) \wedge$
$\exists x_1, x_2, \ldots, x_n(x_1 = z \wedge x_n = y \wedge \forall j = 2,3 \ldots, n-1\ (\texttt{posCauseOf}(x_{j-1}, x_j) \wedge \neg\texttt{Fault}(x_j))))$
"x is a Kitamura and Mizoguchi's absolute cause of y if and only if x (positively) causes some fault z and there is a (positive) causal chain of occurrences $x_1, x_2, \ldots, x_n$, none of which is a fault except possibly the first and the last, which starts with z and ends with y."

df17* $\texttt{ultimateCause}(x,y,s)^* \Longleftrightarrow (\texttt{externalTo}(x,s) \wedge \exists z(\texttt{Fault}(z) \wedge$
$\texttt{posCauseOf}(x,z) \wedge \exists x_1, x_2, \ldots, x_n(x_1 = z \wedge x_n = y \wedge \forall j = 2,3 \ldots, n$
$(\texttt{posCauseOf}(x_{j-1}, x_j) \wedge \texttt{internalTo}(x_j, s))))$
"x is a Kitamura and Mizoguchi's ultimate cause of y with respect to the system s if and only if x is external to s and (positively) causes some fault z and there is a (positive) causal chain of occurrences $x_1, x_2, \ldots, x_n$, all internal to s except for the first, which starts with z and ends with y."

The previous definitions correspond to the concepts in [24] that we are interested the most in. Some of their aspects are arguable, for example, we did not specify if the effect in (df15*) must be an abnormal state or a fault, nor specified properties of the occurrences in the chain in (df16*), while, originally, were described as propagation events.

5. Conclusion

In our work we have carried out a literature review regarding failure-related happenings, selecting some sources from some engineering domains, and discussing and comparing them in depth. Then, exploiting an already-existing causal ontology [25], we propose a formal taxonomy of failure-related happenings, and, finally, we discuss its relation with similar works present in the engineering literature. Such relations are only informally described, but this is unavoidable as the other works considered are not expressed in formal languages.

The main result is a revision that proposes an overarching model clarifying and putting together consolidated knowledge in the field of failure analysis. In particular, a formal serialization has been made available and explained, with the hopeful expectation that such an effort contributes to deepening the understanding of failure analysis. Additionally, the developed formal ontology will expectantly allow, in planned future work, to represent causal explanations of malfunctions (e.g. the instances of the activity of root cause analysis) to be represented as models of the theory, so that reasoning can be used to check if a root cause analysis complies with the ontology, or to augment the analysis by automated inferences.

We are especially indebted to Riichiro Mizoguchi, for his valuable inputs and discussions about causation and malfunctioning, in particular for illustration and explanation of the causal ontology described in [25].

[13] Notice that in the following formulas we assume that there is a `externalTo` relation analogous to `internalTo` and both of these admit items and occurrences as arguments. Note also that definitions (df16*) and (df17*) are definitions in second-order logic.

This work was supported by the projects Ontology-driven data documentation for Industry Commons (OntoCommons), Future AI Research (FAIR) – PE00000013; PRIN 2022 "I-TROPHYTS" – grant. n. 20224TAETP; and PRIN 2022 PNRR "SORTT" – grant. n. P2022H74YP.

References

[1] C. Matthews, A Practical Guide to Engineering Failure Investigation: Matthews/A Practical Guide to Engineering Failure Investigation, John Wiley & Sons, Ltd, Chichester, UK, 1998.
[2] International Electrotechnical Vocabulary (IEV) - Part 192: Dependability, Standard, International Electrotechnical Commission, Geneva, CH (Feb. 2015).
[3] L. Del Frate, M. Franssen, P. Vermaas, Towards a trans-disciplinary concept of failure for integrated product development, International Journal of Product Development 14 (1-4) (2011) 72–95. doi:10.1504/IJPD.2011.042294.
[4] L. Del Frate, Preliminaries to a formal ontology of failure of engineering artifacts, in: M. Donnelly, G. Guizzardi (Eds.), Formal Ontology in Information Systems: Proceedings of the Seventh International Conference (FOIS 2012), IOS Press, Netherlands, 2012, pp. 117–130, null ; Conference date: 24-07-2012 Through 27-07-2012.
[5] M. Rausand, K. Øien, The basic concepts of failure analysis, Reliability Engineering & System Safety 53 (1) (1996) 73–83.
[6] M. Rausand, A. Barros, A. Høyland, System Reliability Theory: Models, Statistical Methods, and Applications, third edition Edition, Wiley Series in Probability and Statistics, John Wiley & Sons, Inc, Hoboken, NJ, 2020.
[7] 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 1 (1) (2004) 11–33.
[8] W. E. Vesely, F. Goldberg, N. H. Roberts, D. Haasl, Fault Tree Handbook, 1987.
[9] Maintenance - Maintenance terminology, Standard EN 13306, Cen-en (2017).
[10] C. Woods, M. Selway, M. Hodkiewicz, F. Ameri, M. Stumptner, W. Sobel, On the notion of maintenance state for industrial assets, in: CEUR Workshop Proceedings, Vol. 2969, 2021.
[11] L. R. Baker, Philosophy Documentation Center, The Metaphysics of Malfunction, Techné: Research in Philosophy and Technology 13 (2) (2009) 82–92.
[12] L. Jansen, Functions, Malfunctioning, and Negative Causation, in: A. Christian, D. Hommen, N. Retzlaff, G. Schurz (Eds.), Philosophy of Science, Vol. 9, Springer International Publishing, Cham, 2018, pp. 117–135.
[13] R. Mizoguchi, Y. Kitamura, S. Borgo, Towards A Unified Definition of Function, Frontiers in Artificial Intelligence and Applications (2012) 15.
[14] L. Vieu, S. Borgo, C. Masolo, Artefacts and Roles: Modelling Strategies in a Multiplicative Ontology, in: Frontiers in Artificial Intelligence and Applications, 2008, p. 15.
[15] D. A. Spear, C. Werner, S. Barry, Functions in Basic Formal Ontology, Applied Ontology 11 (2) (2016) 103–128.
[16] MIL-STD-1629A: Procedures for performing a failure mode, effects and criticality analysis, Tech. rep., Department of Defence, Washington DC (24 November1980).
[17] Military handbook electronic reliability design handbook, Tech. rep.
[18] Failure Mode/Mechanism Distributions 1991, Tech. rep., Reliability Analysis Center, Rome, NY (1991).
[19] USAAMRDL technical report 71-74 helicopter development reliability test requirements, Tech. rep., Eustis directorate (1972).
[20] Selskapet for Industriell og Teknisk Forskning (Ed.), OREDA: Offshore Reliability Data Handbook, 3rd Edition, Norske Veritas, Høvik, 1997.
[21] W. E. Vesely, F. Goldberg, N. H. Roberts, D. Haasl, Fault Tree Handbook, 1987.
[22] L. Del Frate, S. D. Zwart, P. A. Kroes, Root cause as a u-turn, Engineering Failure Analysis 18 (2) (2011) 747–758, the Fourth International Conference on Engineering Failure Analysis Part 1. doi:https://doi.org/10.1016/j.engfailanal.2010.12.006.
URL https://www.sciencedirect.com/science/article/pii/S1350630710002384
[23] Y. Koji, Y. Kitamura, R. Mizoguchi, TOWARDS MODELING DESIGN RATIONAL OF SUPPLEMENTARY FUNCTIONS IN CONCEPTUAL DESIGN, in: Proceedings of the TMCE 2004, 2004, p. 14.
[24] Y. Kitamura, R. Mizoguchi, An Ontological Analysis of Faults (1999) 8.
[25] F. Toyoshima, R. Mizoguchi, M. Ikeda, Causation: A functional perspective, Applied Ontology 14 (1) (2019) 43–78.
[26] ISO 24707, Information technology — Common Logic (CL) — A framework for a family of logic-based languages (2018).
[27] F. Rodrigues, M. Abel, What to consider about events: A survey on the ontology of occurrents, Applied Ontology 14 (2019) 1–36. doi:10.3233/AO-190217.
[28] F. Compagno, S. Borgo, Towards a formal ontology of engineering functions, behaviours, and capabilities, Submitted for Publication to the Semantic Web Journal (2022).
[29] K. Blache, A. Shrivastava, Defining failure of manufacturing machinery and equipment, in: Proceedings of Annual Reliability and Maintainability Symposium (RAMS), 1994, pp. 69–75. doi:10.1109/RAMS.1994.291084.
[30] R. Mizoguchi, A solution to the problem of causation (2023).
URL https://arxiv.org/ftp/arxiv/papers/2307/2307.07517.pdf