






# WATCHDOG: an ontology-aWare risk Assessment approach via object-oriented DisruptiOn Graphs<sup>★</sup>

Stefano M. Nicoletti<sup>1</sup>, E. Moritz Hahn<sup>1</sup>, Mattia Fumagalli<sup>2</sup>,  
Giancarlo Guizzardi<sup>1</sup>, and Mariëlle Stoelinga<sup>1,3</sup>

<sup>1</sup> University of Twente, Enschede, the Netherlands

<sup>2</sup> Free University of Bozen-Bolzano, Bozen, Italy

<sup>3</sup> Radboud University, Nijmegen, the Netherlands

{s.m.nicoletti,e.m.hahn,g.guizzardi,m.i.a.stoelinga}@utwente.nl,  
mattia.fumagalli@unibz.it

**Abstract** When considering risky events or actions, we must not downplay the role of involved objects: a charged battery in our phone averts the risk of being stranded in the desert after a flat tyre, and a functional firewall mitigates the risk of a hacker intruding the network. The *Common Ontology of Value and Risk* (COVER) highlights how the role of objects and their relationships remains pivotal to performing transparent, complete and accountable risk assessment. In this paper, we operationalize some of the notions proposed by COVER – such as *part-hood* between objects and *participation* of objects in events/actions – by presenting a new framework for risk assessment: WATCHDOG. WATCHDOG enriches the expressivity of vetted formal models for risk – i.e., *fault trees* and *attack trees* – by bridging the disciplines of *ontology* and *formal methods* into an ontology-aware formal framework composed by a more expressive modelling formalism, *Object-Oriented Disruption Graphs* (DOGs), logic (DOGLog) and an intermediate query language (DOGLang). With these, WATCHDOG allows risk assessors to pose questions about *disruption propagation*, *disruption likelihood* and *risk levels*, keeping the fundamental role of objects at risk always in sight.

**Keywords:** ontology, logic, risk, COVER, fault trees, attack trees

## 1 Introduction

Risk assessment is a key activity to identify, analyze and prioritize the risk in a system, and come up with (cost-)effective countermeasures [38]. This is true when considering *safety* (i.e., the absence of risk connected with unintentional malfunctions) and *security* (i.e., the absence of risk linked with intentional attacks) [35]. To perform transparent, complete and accountable risk assessment, it is fundamental to explicitly account for the role objects play in *Events* (including *Actions*) in which they participate, and for how their status affects safety and security interplay: a door being locked causes the impossible escape event in

<sup>★</sup>This work was partially funded by the NWO grant NWA.1160.18.238 (PrimaVera), the EU’s Horizon 2020 Marie Curie grant No 101008233, ERC Consolidator and Proof of Concept Grants 864075 (*CAESAR*) and 101187945 (*RUBICON*).

case of fire but simultaneously stops the action of a burglar entering your house [23, 27, 36]. Formalisms widely employed in industry and academia to conduct risk assessment – such as *fault trees* [31] and *attack trees* [34] – are not equipped to explicitly reason about objects. Without an explicit representation of objects at risk, it is impossible to evaluate their role in risk scenarios and to correctly evaluate the overall risk imposed on these objects: e.g., what is the risk level my car is subject to, considering that both its tyres can break (safety-related event) and its onboard computer can be hacked (security-related action)?

Our objective here is to provide an ontology-grounded formal approach for object-based risk representation and reasoning by combining and extending standard formalisms for safety (*fault trees*) and security (*attack trees*). To address this lack of expressivity, two promising fields must be taken into account: *ontologies for risk* and *model-based risk assessment*. On the one hand, risk ontologies – like the *Common Ontology of Value and Risk* (COVER) [33] – excel in providing a structured ground for reasoning about a specific domain of knowledge, transparently and explicitly laying out key concepts and relationships needed to reason about risk. While excellent for conceptualization and transparency, ontologies are however not designed to enable quantitative and applied risk evaluations. On the other hand, specific model-based technologies from the field of formal methods – like *fault trees* (FTs) and *attack trees* (ATs) – excel in providing applicable, tried and tested instruments for rigorous and quantitative risk assessment. These methods, however, sometimes rely on opaque conceptual assumptions and, in particular, do not offer the expressivity needed to explicitly reason about objects at risk. With WATCHDOG we propose a risk assessment framework that enriches and extends the expressivity of vetted model-based technologies – such as FTs and ATs – while grounding them in the conceptual clarity of COVER.

Fundamental elements highlighted by the COVER ontological framework [33] – such as the *participation* of a given object in a risk-related *action/event* or the *parthood* relationship between different objects – are not expressible in classical risk assessment formalisms, such as FTs and ATs. We address this gap and operationalize these concepts by presenting *object-oriented DisruptiOn Graphs* (DOGs), a new formalism that extends the strengths of classical FT- and AT-based risk analysis accounting for the role of objects at risk (OaRs) in *disruption propagation*, *likelihood* and *risk calculation*<sup>†</sup>. Moreover, to perform transparent decision-making w.r.t. safety and security of systems, practitioners need the ability to analyse their models in a meaningful and thorough way. To cater for this need, we present DOGLog – a logic to formally query DOGs – and DOGLang, an intermediate domain-specific language to ease the querying process. With DOGLog and DOGLang practitioners can query DOGs to learn meaningful information about systems *disruptions* and *risk levels*. One could ask, for example: Given that one of my tyres breaks, is my entire field trip compromised? Is the probability of an attacker compromising the network larger/smaller than  $p$ ?

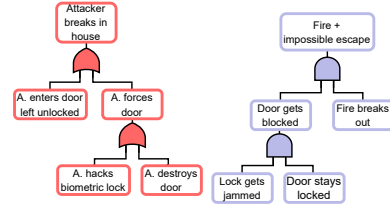
<sup>†</sup> We adopt these notions from previous work. For a more in-depth discussion on *risk*, *disruption*, *propagation*, and their relation with *risk propagation*, see [12, 35].

Given an object at risk (e.g., my laptop), what is the most risky security Action/safety Event in which it participates? What is the maximum risk level imposed on my laptop, given all the Actions/Events in which it participates? Finally, we showcase property specification in DOGLog and DOGLang on an *DOG* model for a variant of small but well-known and representative example from safety-security literature, modelling safety and security risks on a household given the status of a door lock [23, 27, 36].

## 2 Baseline Research

Fault Trees (FTs) and Attack Trees (ATs) constitute a sensible starting point as they are popular technologies that already encode key concepts highlighted in the *Common Ontology of Value and Risk* (COVER) – such as *Events* and *Actions* – and pose a solid ground for model-based risk assessment. Fault tree analysis (FTA) [31] is a widespread technique to support safety risk assessment, and the use of fault trees is required, e.g., by the Federal Aviation Administration, the Nuclear Regulatory Commission, in the ISO 26262 standard [17] for autonomous driving and for software development in aerospace systems. A *fault tree* (FT) (see Fig. 1, right) models *Events* that describe how component failures arise, and propagate disruption through the system, eventually leading to system-level failures. Leaves in a FT represent *basic events* (BEs), i.e. elements of the tree that do not need further refinement. Once these fail, the failure is propagated through the *intermediate events* (IEs) via *gates*, to eventually reach the *top level event* (TLE), which symbolizes system failure. When considering model-based risk assessment of systems security – *attack trees* (ATs) are widely employed. ATs (see Fig. 1, left) are hierarchical diagrams that represent malicious *Actions* that can lead to a system being compromised [34, 24]. ATs are referred to by many system engineering frameworks, e.g. *UMLsec* [21] and *SysMLsec* [30], and are supported by industrial tools such as Isograph’s *AttackTree* [19]. The TLE of an AT represents the attacker compromising the entire system, and the leaves represent *basic attack steps* (BASes): actions of the attacker that can no longer be refined. As for FTs, intermediate nodes in ATs are labelled with gates.

The *Common Ontology of Value and Risk* (COVER) [33] is based on *UFO* [16] – a foundational ontology. It embeds a domain-independent conceptualization of risk, has been subject to validation and proper comparison to the literature of risk in risk analysis and management at large (e.g. [18]), and it is built upon widespread definitions of risk. This ontology has already shown its utility in constructing formalisms for risk quantification and propagation [12], and embeds several key assumptions about the nature of risk, which align with those in



**Figure 1:** An attack tree (left) and a fault tree (right) for the locked door example.

the literature in risk assessment. Firstly, risk is **experiential**. This means that the notions of “event” and “object” are deeply entangled and, when assessing the risk an object is exposed to, one aggregates risks ascribed to events that can impact the object. For instance, consider the risks your laptop is exposed to. To assess them, you will need to consider: 1. which of your goals depend on your laptop (e.g. work deliverables); 2. what can happen to your laptop such that it would hinder its capability to achieve your goals (e.g. its screen breaking); 3. which other events could cause these (e.g. you dropping it on the floor). Then the risk your laptop (object) is exposed to is the aggregation of the risk of it falling (event) and breaking (event), and so on. The second assumption is that risk is **contextual**. Thus, the magnitude of the risk an object is exposed to may vary even if all its intrinsic properties (e.g., vulnerabilities or states) stay the same. To exemplify, let us pick one risk event involving your house door, namely that of robbers breaking into it. Naturally, the properties of the door – such as having a strengthened blocking mechanism – influence the magnitude of this risk. Still, the tools used by the robbers can significantly increase how risky the breaking event is. Lastly, another assumption that we derive from COVER is that risk is grounded on **uncertainty** about events and their outcomes. This is a very standard position – see [18] and [3] – which implies that likelihood is positively correlated with how risky an event is. For instance, the risk of encountering a bear is higher while walking in a forest than in an urban park, simply because it is more likely in the former case.

In COVER, a RISK EXPERIENCE is a multifaceted hypothetical occurrence that can be broken down into RISK EVENTS, further categorized into THREAT EVENTS and LOSS EVENTS. THREAT EVENTS are hypothetical occurrences capable of precipitating LOSS EVENTS, which, in turn, are incidents that undermine the objectives of a RISK SUBJECT, the AGENT whose viewpoint is under scrutiny in the risk evaluation process. A LOSS EVENT might involve OBJECTS AT RISK and RISK ENABLERS. Dispositions of objects at risk and risk enablers that can be manifested as threat and loss events are VULNERABILITIES. As discussed in [33] COVER accounts also for a numerical evaluation of RISK linked to a RISK ASSESSMENT. We emphasise here that the quantification of *risk* is applicable solely to *anticipated scenarios* that have the potential (though not certainty) to materialize. The ontology tackles this concern by acknowledging the feasibility of anticipated events, as discussed in [13].

In order to ground WATCHDOG in COVER, we make a number design assumptions, which are used in the construction of both the new proposed model (DOGs) and logic (DOGLog). We further discuss how they relate to – and are grounded in – the COVER ontology, their implications and limitations. Throughout the paper, we highlight them with the **assumption** tag whenever they play an active role. We distinguish between two types of assumptions: *operational* and *structural*. Operational assumptions introduce constraints that are stricter than necessary due to the novel nature of this work, requiring a cautious and incremental approach. Structural assumptions, on the other hand, directly derive from the existing COVER ontology as-is. We elaborate on these in the sequel:

- **Assumption 1** (*structural*): the attribution of impact on parent elements in the *DOG* is independent of the attribution on children elements. COVER grounds this assumption by clearly distinguishing the notions of *probability*, *impact*, and *risk*. On DOGs we propagate disruption (i.e., attacks/failures propagation in the system) and – quantitatively – their probability values. In the case of disruption likelihood, the values assigned to each event depend on those assigned to the other events to which they are related. Differently, impact values are assigned independently to each single event. For example, the probability value assigned to a “breaking laptop” event naturally depends on the probability value of a possible related event such as “stumbling”. This is not the case for what concerns the loss value (or associated impact) of the two events (in itself, stumbling may not be a problem, while laptop breaking represents something serious). Note that by distinguishing between probability and impact values, we enable a clearer assessment of an event’s impact, independent of the likelihood of its occurrence;
- **Assumption 2** (*structural*): OaRs that can participate in parent elements of an *DOG* are a collection of all OaRs that can participate in their children elements, plus additional OaRs added by the risk assessor. This assumption is in line with the representation of events in COVER. In this context, events modelled as children of other events in the graph can be naturally taken as parts of the parent event, and in a simplified view, this allows inferring that objects that participate in parts (or sub-events) of an event, also participate in the event itself;
- **assumption 3** (*operational*): for each *OaR* that participates in an element (event/action) of the *DOG*, we assume that its parts participate in it as well, but the opposite does *not* hold. For instance, if *Door* participates in *Door stays locked* also its part – namely, *Lock* – participates in it, but if *Lock* participates in *Lock breaking* this does not imply that *Door* participates in that event. This is, again, aligned with the theory about events, objects and their parts encoded by UFO and inherited by COVER<sup>‡</sup>;
- **assumption 4** (*operational*): in computing risk values, we assume the attacker already knows which failure occurred in the system before acting. Looking at the conceptualization provided in COVER, this aligns with the composition of the “risk experience” concept, which not only accounts for the role of “passive” elements involved in the assessment of risk (e.g., “object at risk”) but also for the role of the “active” elements involved (see, for instance, the concept of “threat object” and related *threat capability*). This, then, supports the two-step representation of the assessment process we propose, where, firstly, vulnerabilities in a system are identified and, secondly, based on these, threats can be activated. This solution allows for simulations that act as operationalizations of the concept of “risk experience”;
- **assumption 5** (*operational*): the attacker can adapt its strategy to the event under consideration. E.g., when computing max total risk (Sec. 4) we assume the attacker can maximise the risk level for each individual event in

<sup>‡</sup>For a more detailed focus on this assumption we refer the reader to [15, 14].





*Locked?* 4. What is the minimal risk level associated with the OaR *Door*, given all the events/actions in which it participates? In [Sec. 5](#) we will formalize these queries via our logic (DOGLog) and query language (DOGLang).

**Definition 1 (Object-Oriented Disruption Graph).** An Object-Oriented Disruption Graph (DOG)  $G$  is a tuple  $(A, F, O, B)$  where  $A$  is an attack tree,  $F$  is a fault tree,  $O$  is an object graph and  $B$  is a disruption knowledge base. As mentioned in [\[35\]](#), ATs and FTs can be syntactically unified under the *disruption tree* (DT) model:

**Definition 2 (Disruption Tree).** A disruption tree (DT)  $T$  is a tuple  $(N, E, t)$  where  $(N, E)$  is a rooted directed acyclic graph, and  $t: N \rightarrow \{\text{OR}, \text{AND}, \text{LEAF}\}$  is a function s.t. for  $v \in N$ , it holds that  $t(v) = \text{LEAF}$  iff  $v$  is a leaf. Moreover,  $ch: N \rightarrow 2^N$  gives the set of children of a node and  $T$  has a unique root, i.e.,  $R_T$ .

We also define the set of intermediate events  $\text{IE} = N \setminus \text{LEAF}$ . Moreover, if  $u \in ch(v)$  then  $u$  is called a *child* of  $v$ , and  $v$  is a *parent* of  $u$ . Furthermore, we employ only AND- and OR-gates in the AT/FT components of the model. The behaviour of a DT  $T$  can be expressed through its *structure function* [\[31\]](#) -  $f_T$ : if we assume the convention that a LEAF has value 1 if disrupted and 0 if operational, the structure function indicates the status of the root node – or *top level event* (TLE) – given the status of all the LEAVES of  $T$ . Thus, for each set of LEAVES we can identify its characteristic vector  $\vec{b}$ : we refer to this vector as a *scenario*. We denote by  $\mathcal{S}_T = 2^{\text{LEAF}_T}$  the universe of scenarios of  $T$ . When further distinction is needed between ATs and FTs constructs, we use (respectively) the subscripts  $\_A$  and  $\_F$ : e.g., we refer to a scenario on an AT (resp. FT) as an *attack scenario* (resp. *fault scenario*), represented by  $\vec{b}_A$  (resp.  $\vec{b}_F$ ). As shown before, in FT- and AT-related literature nodes canonically represent respectively *events* and *attack steps*: one might easily map *attack steps* and *events* to the terminology chosen in the COVER ontology, for which nodes  $N_A$  of an AT represent *Actions* and nodes  $N_F$  of a FT represent *Events*. From this point on, we will use the general term *elements* to refer indistinctly to nodes in FTs and ATs. To enrich ATs and FTs, we introduce *Objects at Risk* (OaRs) that explicitly capture impacted objects in (safety and security) risk experiences.

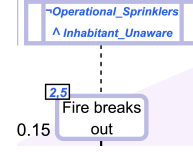
**Definition 3 (Object Graph).** An object graph (OG)  $O$  is a rooted directed acyclic graph  $(N_O, E_O, OP, cOP)$  where: 1. nodes in  $N_O$  represent Objects at Risk (OaRs); 2. directed edges in  $E_O \subseteq N_O \times N_O$  represent the parthood relation between OaRs; 3. properties on OaRs are atomic propositions  $op \in OP$ ; and 4.  $cOP: N_O \rightarrow 2^{OP}$  returns a set of atomic propositions of a node  $v \in N_O$ .

Moreover,  $ch: N_O \rightarrow 2^{N_O}$  gives the set of *parts* of a node and  $O$  has a unique root, denoted  $R_O$ . As previously hinted, OaRs and the object graph (OG) are represented by connected blue rectangles (see [Fig. 2, page 6](#)). Similarly to DTs' evaluation, we need a way to evaluate properties of OaRs, thus:

**Definition 4 (Evaluating Properties of OaRs).** We let a configuration  $\vec{b}_O$  be the Boolean vector assigning values to properties of OaRs in  $OP$  and we let  $f_O: \mathbb{B}^n \times OP \rightarrow \mathbb{B}$  be a valuation such that  $f_O(\vec{b}_O, op) = 1$  iff the Boolean value of a property  $op \in OP$  equals 1 given  $\vec{b}_O$ .

Furthermore, we let  $\mathcal{C}$  be the set of all possible configurations. Finally, we introduce a *Disruption Knowledge Base* (DKB) that establishes a formal relation between elements in ATs and FTs and OaRs that can participate in them. We also define attribution of an impact value to ATs and FTs elements and their *preconditions* and *conditions* in the DKB. *Preconditions* are relevant properties of OaRs that can participate in a given event/action. These properties must be arranged in a specific way for events/actions to happen: *conditions* express this arrangement.

*Example 1.* Consider the excerpt of the *DOG* of Fig. 2 in Fig. 3: conditions for the event *Fire breaks out* – in blue, connected with a dashed line – are encoded in the Boolean formula  $\neg \text{Operational\_Sprinklers} \wedge \text{Inhabitant\_Unaware}$ : in fact, the OaRs *Smart house* and *Inhabitant* participate in the event *Fire breaks out* and *Operational\\_Sprinklers* and *Inhabitant\\_Unaware* are the relevant properties of these participating objects at risk. These are exactly the *preconditions* for event *Fire breaks out* and they must be set resp. to *false* and *true* in conjunction for *Fire breaks out* to happen.



**Figure 3:** An excerpt of Fig. 2.

**Definition 5 (*Disruption Knowledge Base*).** A disruption knowledge base (DKB)  $B$  is a tuple  $(D, Im, Pa, Pr)$  where:

1.  $D = N_A \cup N_F \cup N_O$  is an entity domain where  $N_A, N_F$  and  $N_O$  are pairwise disjoint
2.  $Im: N_A \cup N_F \rightarrow \mathbb{R}_{\geq 0}$  is a function that returns an impact factor for each element  $v \in N_A \cup N_F$
3.  $Pa: N_A \cup N_F \rightarrow 2^{N_O}$  is a function that for each element  $v \in N_A \cup N_F$  returns a set of OaRs that can participate in  $v$
4. For each element  $v \in N_A \cup N_F$ ,  $Pr: N_A \cup N_F \rightarrow 2^{OP}$  is a function that returns the set of its preconditions  $Pr(v) = cOP(o_1) \cup cOP(o_2) \dots \cup cOP(o_n)$  with  $o_i \in Pa(v)$
5. For each element  $v \in N_A \cup N_F$ , conditions on  $v$  are represented by a Boolean formula  $Cond(v)$  over its preconditions  $Pr(v)$

Do note that *conditions* naturally map to *descriptions of situations* in COVER.

**Assumption 1:** Moreover, we assume that attribution of impact on parent elements (see item 2) is independent of the attribution on children elements.

**Assumption 2:** We also assume that OaRs that can participate in parent elements (events/actions) – being decorated with AND-gates or OR-gates in the DT – are a collection of all OaRs that can participate in their children elements, plus additional OaRs added by the risk assessor (exactly what the  $Pa$  function returns). As seen with properties of OaRs, we need a way to evaluate whether conditions on a given DT element (it being an action or an event) are satisfied:

**Definition 6 (*Evaluating Conditions of Elements*).** We let  $f_{Cond}: \mathbb{B}^n \times Form \rightarrow \mathbb{B}$  be the evaluation function for conditions – with  $Form$  being the set of Boolean formulae – such that given a configuration  $\vec{b}_O$  and conditions  $Cond(v)$  for an element  $v$ ,  $f_{Cond}(\vec{b}_O, Cond(v)) = 1$  iff the Boolean assignment in  $\vec{b}_O$  satisfies  $Cond(v)$ .



To summarize: for each element  $v$  we construct a Boolean formula  $Cond(v)$  over the set of preconditions  $Pr(v)$ , which are exactly the relevant *properties* of OaRs that *participate* in  $v$  which are needed for  $v$  to happen. To do so, we collect the  $n$  OaRs that can participate in  $v$  with  $Pa(v)$  and – for each of these objects  $o_i \in \{o_1, \dots, o_n\}$  – we collect its *properties* via  $cOP(o_i)$ . **Assumption 3:** For each OaR that participates in element (event/action)  $v$ , we assume that its parts participate in  $v$  as well, but the opposite does *not* hold. The union of all these collected sets  $cOP(o_i)$  is exactly the set  $Pr(v)$  of *preconditions* on  $v$ . The Boolean formula  $Cond(v)$  over preconditions in  $Pr(v)$  for  $v$  represents its *conditions*. It is important to note that, in a practical setting, a user would iteratively be asked whether 1. all participating objects in  $v$  and 2. all preconditions of a given participating OaR are relevant for conditions on  $v$ . The ability to isolate and compute separately both preconditions and conditions will be functional in this setting. E.g., the property *Lock\_Locked* is relevant for preconditions of an element  $v = \text{Attacker enters door left unlocked}$ , but *Lock\_Hackable* is not: thus, it is not included as an atom in  $Cond(v)$ . As hinted, understanding computations on DOGs requires further attention on the interplay between Actions (resp. Events) in ATs (resp. FTs) and their conditions. Since  $Cond(v)$  is a Boolean formula over preconditions for  $v \in N_A$  (resp.  $v \in N_F$ ), for  $v$  to be attacked (to fail) the entire Boolean formula composed by  $v \wedge Cond(v)$  must evaluate to *true*. To account for this, let us define an extended structure function for DTs where, for  $v$  to be disrupted it would be necessary to have  $f_T^\circ(\vec{b}, \vec{b}_O, v)$  return 1, i.e., both the node in question should be disrupted and its conditions must be satisfied.

**Definition 7 (Extended Structure Function).** *The extended structure function of a disruption tree  $T$  is a function  $f_T^\circ: \mathbb{B}^n \times \mathbb{B}^n \times N \rightarrow \mathbb{B}$  that takes as input a scenario  $\vec{b}$ , a configuration  $\vec{b}_O$  and an arbitrary element  $v \in N$ . We define it as follows:*

$$f_T^\circ(\vec{b}, \vec{b}_O, v) = \begin{cases} b_i \wedge f_{Cond}(\vec{b}_O, Cond(v)) & \text{if } v = v_i \in \text{LEAF} \\ \bigvee_{v' \in ch(v)} f_T^\circ(\vec{b}, \vec{b}_O, v') \wedge f_{Cond}(\vec{b}_O, Cond(v)) & \text{if } v \in \text{IE and } t(v) = \text{OR} \\ \bigwedge_{v' \in ch(v)} f_T^\circ(\vec{b}, \vec{b}_O, v') \wedge f_{Cond}(\vec{b}_O, Cond(v)) & \text{if } v \in \text{IE and } t(v) = \text{AND} \end{cases}$$

## 4 DOGLog: a Logic to reason about DOGs

We construct our logic on three syntactic layers, represented with  $\phi$ ,  $\psi$  and  $\xi$ . Layer 1 formulae reason about disruption propagation: the atomic propositions  $a$  in DOGLog can represent any element in an AT or FT, and any property of OaRs, i.e.,  $a \in N_A \cup N_F \cup OP$ . Formulae can be combined through usual Boolean connectives. Furthermore, we can set evidence to construct what-if scenarios:  $\phi[a \mapsto \text{bool}]$  sets the element  $a$  in  $\phi$  to either 0 or 1, representing an event/action taking place or an object property being *true* or *false*. Finally, DOGLog allows reasoning about *minimal risk scenarios* (MRSs): minimal assignments on leaves of the FT and AT, such that a formula is satisfied (e.g., such that an event/action takes place). Note that MRSs are always evaluated by fixing a specific configuration, i.e. a specific status of object properties. Layer 2 formulae reason about

disruption propagation probabilities. We can check whether the disruption probability of a given  $\phi$  formula (e.g., a given event/action) is bounded by a selected threshold – with our comparison operator  $\bowtie \in \{<, \leq, =, \geq, >\}$  – and we can also set probabilistic evidence, i.e., formulate scenarios where an event/action  $e \in N_A \cup N_F$  is assigned a specific disruption probability  $q$ . Combining layer 2 formulae with Boolean operators is also allowed. Lastly, layer 3 reasons about safety- and security-related risk levels. One can ask what are the most risky actions/events in which an OaR participates – with  $*$   $\in \{A, F\}$  symbolizing resp. AT and FT nodes, i.e., actions and events. Moreover, one can ask what is the max/min total risk level to which an OaR is subject, aggregating risk from both safety- and security-related events/actions in which it participates, and what is the optimal configuration of object properties to minimise risk on an OaR. Finally, one can also set evidence on object properties, i.e., forcefully set them to *true* or *false* to create insightful what-if scenarios. Note that when setting evidence we usually assign values to  $a \in \text{LEAVEs} \cup \text{OP}$  in layer 1, and to  $e \in \text{LEAVEs}$  in layer 2. We can however assign values to IEs of ATs and FTs if 1.  $a/e \in N_A \cup N_F$  is a module [10], i.e., all paths between descendants of  $a/e$  and the rest of the AT or FT pass through  $a/e$  2. and none of the descendants of  $a/e$  are present in the formula. If so, we prune that (sub)AT or (sub)FT (and relative conditions) and treat occurring IEs as LEAVEs. We can formally define the syntax for DOGLog as follows:

Layer 1:  $\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi[a \mapsto \text{bool}] \mid \text{MRS}(\phi)$   
 Layer 2:  $\psi ::= \text{P}(\phi) \bowtie p \mid \neg\psi \mid \psi \wedge \psi \mid \psi[e \mapsto q]$   
 Layer 3:  $\xi ::= \text{MostRisky}_*(o) \mid \underset{\max}{\text{TotalRisk}}(o) \mid \underset{\min}{\text{TotalRisk}}(o) \mid \text{OptimalConf}(o) \mid \xi[\text{op} \mapsto \text{bool}]$

## 5 Object-Oriented Risk Queries: DOGLog and DOGLang

To ease the usability of our logic, we present DOGLang, a Domain Specific Language (DSL) for DOGLog. Defining languages and tools to specify properties and requirements is common: in [9] the authors capture high-level requirements for a steam boiler system in a human-readable form with SADL. Further controlled natural languages for knowledge representation include Processable English (PENG) [37], Controlled English to Logic Translation (CELT) [28], Computer Processable Language (CPL) [7] and FRETish [8]. DOGLang is constructed by adhering to the same design philosophy of *LangPFL* – a domain specific language for FTs that was developed in the literature [25]. As for *LangPFL*, DOGLang is inspired by the aforementioned languages for their ease of use and close proximity to natural language. DOGLang expresses only a fragment of DOGLog. Notably, nesting of formulae is disallowed: we retain most of the expressiveness of DOGLog while making property specification easier. In DOGLang, *DOG* elements are referred to with their short label and each operator in DOGLog has a counterpart in the DSL: Boolean operators, *not*, *and*, *or*, *impl*...; setting the value of *DOG* elements to Boolean or probabilistic values, *set*, *set\_prob*; minimal risk scenarios MRSs, *MRS*[...]; operators to check disruption probability thresholds, *Prob*[...]  $\bowtie$  ... (note that  $\bowtie \in \{<, \leq, =, \geq, >\}$ ); and to reason about risk levels aggregated on a given object and about risky actions/events in which this object participates, *MostRiskyA*[...], *MostRiskyF*[...],

Natural Language	Property in DOGLog	DOGLang
Given that an <i>Attacker destroys the door</i> and that the <i>Fire does not break out</i> , are any of the two TLEs happening?	$TLE1 \vee TLE2$ [ $ADD \mapsto 1, FBO \mapsto 0$ ]	<b>assume:</b> set $ADD = 1$ set $FBO = 0$ <b>check:</b> TLE1 or TLE2
What are all the MRSs such that both <i>Loss Events</i> happen, given that <i>Lock is Locked</i> and <i>Door is Frail</i> ?	$\llbracket TLE1 \wedge TLE2 \rrbracket$ [ $LiL \mapsto 1, DiF \mapsto 1$ ]. $\#$	<b>assume:</b> set $LiL = 1$ set $DiF = 1$ <b>computeall:</b> MRS[TLE1 and TLE2]
Is the probability of both successfully forcing the door and fire breaking out lower than 0.05?	$\text{Prob}(AFD \wedge FBO) < 0.05$	<b>assume:</b> <b>check:</b> $\text{Prob}[AFD \text{ and } FBO] < 0.05$
What is the most risky event in which <i>Inhabitant</i> participates, assuming that <i>Lock is Locked</i> ?	$\text{MostRisky}_F(\text{Inhab.})[LiL \mapsto 1]$	<b>assume:</b> set $LiL = 1$ <b>computeall:</b> $\text{MostRiskyF}[\text{Inhab.}]$
What is the max risk level associated with the OaR <i>Door</i> , given all the events/actions in which it participates?	$\text{TotalRisk}_{\max}(\text{Door})$	<b>assume:</b> <b>compute:</b> $\text{MaxTotalRisk}[\text{Door}]$
What is the minimum risk level on <i>Door</i> , assuming that OaR <i>Lock</i> exhibits property <i>Lock Hackable</i> ?	$\text{TotalRisk}_{\min}(\text{Door})[LH \mapsto 1]$	<b>assume:</b> set $LH = 1$ <b>compute:</b> $\text{MinTotalRisk}[\text{Door}]$
What are the properties that all OaRs must exhibit, in order to minimise the risk level associated with the object <i>House</i> , assuming we fix that <i>Door is Frail</i> ?	$\text{OptimalConf}(\text{House})[DiF \mapsto 1]$	<b>assume:</b> set $DiF = 1$ <b>computeall:</b> $\text{OptimalConf}[\text{House}]$

**Table 1:** Risk queries for  $G$  (Fig. 2) in natural language, DOGLog and DOGLang.

$\text{MaxTotalRisk}[\dots]$ ,  $\text{MinTotalRisk}[\dots]$ ,  $\text{OptimalConf}[\dots]$ . One can specify properties in DOGLang by utilizing operators inside structured templates. Assumptions on the status of *DOG* elements can be specified under the **assume** keyword. These assumptions will be automatically integrated with the translated formula accordingly, e.g., **set** or **set\_prob** will be translated with the according operators to set evidence, while other assumptions will be the antecedent of an implication. A second keyword separates specified formulae from the assumptions and dictates the desired result: **compute** and **computeall** compute and return desired values, i.e., probability values, and lists of events/actions/configurations and MRSs respectively, while **check** establishes if a specified property holds.

In Table 1 we exemplify some queries on the *DOG* for the smart house locked door example in Fig. 2. These queries exemplify the expressive power enabled by DOGs, DOGLog and DOGLang and are chosen to reflect the different Layers of our logic. It is important to note that syntactically the *DOG* model does not represent answers to these queries right away, so one cannot read them from Fig. 2 directly. Answers are computed as per semantics of both the model and the logic: e.g., the *A. forces door* node is a composite element, and the computation of this complex probability value follows probability composition on the structure of the model in Fig. 2.

## 6 Enabling Risk Computations: DOGLog Semantics

To enable object-oriented risk computations and to ground the meaning of formulae into the enriched model presented in Sec. 3, we define formal semantics for DOGLog. For the first layer of the logic, formulae are evaluated on the fol-

lowing model  $\mathcal{M} = \langle \vec{b}_R, \vec{b}_O, G \rangle$  where a *risk scenario*  $\vec{b}_R = (b_1, \dots, b_k)$  is defined as  $\vec{b}_R = \vec{b}_A \cup \vec{b}_F$ ,  $\vec{b}_O$  is a configuration and  $G$  is an *DOG*. Formally:

$$\begin{aligned}
\mathcal{M} \models a & \quad \text{iff} \quad \begin{cases} \text{with } a \in N_A & f_A^\circ(\vec{b}_A, \vec{b}_O, a) = 1 \\ \text{with } a \in N_F & f_F^\circ(\vec{b}_F, \vec{b}_O, a) = 1 \\ \text{with } a \in OP & f_O(\vec{b}_O, a) = 1 \end{cases} \\
\mathcal{M} \models \neg\phi & \quad \text{iff } \mathcal{M} \not\models \phi \\
\mathcal{M} \models \phi \wedge \phi' & \quad \text{iff } \mathcal{M} \models \phi \text{ and } \mathcal{M} \models \phi' \\
\mathcal{M} \models \phi[a_i \mapsto \text{bool}] & \quad \text{iff} \quad \begin{cases} \text{with } a_i \in N_A \cup N_F & \mathcal{M}' \models \phi \text{ with } \vec{b}_R' = (b'_1, \dots, b'_k) \in \mathcal{M}', \\ & b'_i = \text{bool} \in \mathbb{B} \text{ and } b'_j = b_j \text{ for } j \neq i \\ \text{with } a_i \in OP & \mathcal{M}' \models \phi \text{ with } \vec{b}_O' = (b'_1, \dots, b'_m) \in \mathcal{M}', \\ & b'_i = \text{bool} \in \mathbb{B} \text{ and } b'_j = b_j \text{ for } j \neq i \end{cases} \\
\mathcal{M} \models \text{MRS}(\phi) & \quad \text{iff } \vec{b}_R \in \llbracket \phi \rrbracket_{\mathcal{M}}
\end{aligned}$$

With  $\llbracket \phi \rrbracket_{\mathcal{M}}$  we denote the *minimal satisfaction set* of risk scenarios for  $\phi$ , i.e., the set of minimal risk scenarios  $\vec{b}_R$  that satisfy  $\phi$  given  $G$ . We define  $\llbracket \phi \rrbracket_{\mathcal{M}}$  as follows:  $\llbracket \phi \rrbracket_{\mathcal{M}} = \{ \vec{b}_R \mid \langle \vec{b}_R, \vec{b}_O, G \rangle \models \phi \wedge \nexists \vec{b}_R' \subseteq \vec{b}_R \wedge \langle \vec{b}_R', \vec{b}_O, G \rangle \models \phi \}$ . Note that the set of all minimal risk scenarios for a given  $\phi$  – i.e.,  $\llbracket \phi \rrbracket_{\mathcal{M}}$  – is always computed by fixing a specific configuration  $\vec{b}_O$  first.

Layer two formulae require the introduction of probabilities. First, we need to decorate the leaves of the AT and the FT in  $G$  with probability values. To do so, we let an *attribution on  $G$*  be a map  $\alpha: \text{LEAVES} \rightarrow [0, 1]$ . With a slight abuse of notation, we simply write  $\alpha_G$  for the probability attribution on both the leaves of the AT  $A$  and the FT  $F$  in  $G$ . We then let  $\rho(\phi)$  define the probability of a given layer one formula  $\phi$ . Intuitively: given  $\phi$  and a configuration  $\vec{b}_O$ , we consider every possible fault scenario  $\vec{b}_F \in \mathcal{S}_F$  on  $F$  and how that would impact truth values of FT nodes in  $\phi$ . For each of these fault scenarios, we compute the maximal probability of successfully attacking AT nodes in  $\phi$  under the given configuration  $\vec{b}_O$ . **Assumption 4:** Note that – with this setup – we assume the attacker already knows which FT nodes in  $\phi$  failed. Consequently, we let:

$$\rho(\phi, \vec{b}_O)_{A,F} = \sum_{\vec{b}_F \in \mathcal{S}_F} \text{Prob}(\vec{b}_F) \times \mathbb{P}_A(\text{Set}(\phi, \vec{b}_F, \vec{b}_O))$$

where the probability associated to each fault scenario  $\vec{b}_F \in \mathcal{S}_F$  – with  $v$  as a BE – is calculated via  $\text{Prob}(\vec{b}_F) = \prod_{i=1}^k b_i \times \alpha(v_i) + (1 - b_i) \times (1 - \alpha(v_i))$  and where the maximal probability of successfully attacking  $\phi$  is given by multiplying attributions on BASes in every minimal attack scenario for  $\phi$  –  $\vec{b}_A \in \llbracket \phi \rrbracket_A$  – to then take the maximum between the resulting values of these attacks. Formally:

$$\mathbb{P}_A(\phi) = \max_{\vec{b}_A \in \llbracket \phi \rrbracket_A} \prod_{v \in \vec{b}_A} \alpha(v)$$

This last step is coherent with a more general framework for multiple metric computations on ATs previously defined in [24, 26]. Finally, we account for how every possible fault scenario would impact truth values of atomic propositions

of FT nodes in  $\phi$  by recursively defining  $\text{Set}(\phi, \vec{b}_F, \vec{b}_O)$ , with  $\vec{b}_F \in \mathcal{S}_F$ :

$$\begin{aligned} \text{Set}(a, \vec{b}_F, \vec{b}_O) &= \begin{cases} \text{with } a \in N_A & a \\ \text{with } a \in N_F & \begin{cases} 1 & \text{iff } f_F^\circ(a, \vec{b}_F, \vec{b}_O) = 1 \\ 0 & \text{otherwise} \end{cases} \\ \text{with } a \in OP & \begin{cases} 1 & \text{iff } f_O(a, \vec{b}_O) = 1 \\ 0 & \text{otherwise} \end{cases} \end{cases} \\ \text{Set}(\neg\phi, \vec{b}_F, \vec{b}_O) &= \neg\text{Set}(\phi, \vec{b}_F, \vec{b}_O) \\ \text{Set}(\phi \wedge \phi', \vec{b}_F, \vec{b}_O) &= \text{Set}(\phi, \vec{b}_F, \vec{b}_O) \wedge \text{Set}(\phi', \vec{b}_F, \vec{b}_O) \\ \text{Set}(\phi[a_i \mapsto \text{bool}], \vec{b}_F, \vec{b}_O) &= \text{Set}(\phi, \vec{b}_F, \vec{b}_O)[a_i \mapsto \text{bool}] \\ \text{Set}(\text{MRS}(\phi), \vec{b}_F, \vec{b}_O) &= \text{MRS}(\text{Set}(\phi, \vec{b}_F, \vec{b}_O)) \end{aligned}$$

where 1 and 0 represent the *true* and *false* derived layer 1 formulae. Note that – also due to **Set** – some occurrences can lead to the application of the  $\mathbb{P}_A$  function to either *true* or *false*, i.e., when  $\phi = 1$  or  $\phi = 0$ . In these cases, we fix that  $\mathbb{P}_A(1) = 1$  and  $\mathbb{P}_A(0) = 0$ . With  $\bowtie \in \{<, \leq, =, \geq, >\}$  and an updated model  $\mathbf{M} = \langle \vec{b}_O, G, \alpha_G \rangle$ , semantics for layer two formulae can be defined as follows:

$$\begin{aligned} \mathbf{M} \models \text{P}(\phi) \bowtie p &\text{ iff } \rho(\phi, \vec{b}_O)_{A,F} \bowtie p; & \mathbf{M} \models \neg\psi &\text{ iff } \mathbf{M} \not\models \psi; \\ \mathbf{M} \models \psi \wedge \psi' &\text{ iff } \mathbf{M} \models \psi \text{ and } \mathbf{M} \models \psi'; \\ \mathbf{M} \models \psi[e_i \mapsto q] &\text{ iff } \mathbf{M}(\alpha_G[\text{with } \alpha(v_i) \mapsto q]) \models \psi, \text{ with } v_i \in N_A \cup N_F \end{aligned}$$

Note that the model  $\mathbf{M}$  for layer 2 formulae does not contain a risk scenario  $\vec{b}_R$ : this is because in computing probabilities we already account for both 1. possible fault scenarios  $\vec{b}_F$  and 2. possible attack scenarios  $\vec{b}_A$ . Layer 3 formulae require further attention on OaRs and the participation relation. To compute the risk level associated with an OaR  $o$ , given a certain configuration – e.g., the risk level of *Door*, given that *Lock\_Locked* is set to *false* – we first identify the *set of elements in which  $o$  participates, and for which a satisfying risk scenario plus configuration exist*. We can consider events/actions as layer one atomic formulae  $a$  s.t.  $a \in N_A \cup N_F$ , and – with  $*$   $\in \{\textcolor{red}{A}, \textcolor{blue}{F}\}$  – formally define this set as:

$$(\|o\|)_* = \left\{ a \in N_* \mid o \in Pa(a) \wedge \exists \vec{b}_*, \vec{b}_O. f_*^\circ(\vec{b}_*, \vec{b}_O, a) = 1 \right\}$$

Note that one might want to parameterize some elements of the given configuration  $\vec{b}_O$  to, e.g., compute optimal assignments to minimize risk on a given OaR. To accommodate for this need, we let  $\mathcal{C}_{[op \mapsto \text{bool}]}$  be the set of configurations that could still be compatible with a partial Boolean assignment  $[op \mapsto \text{bool}]$ . E.g.:

*Example 2.* Assume we want to consider only the configurations compatible with setting the evidence that *Lock\_Locked* is *true*, i.e.,  $[\text{Lock\_Locked} \mapsto 1]$  and that we only have two other object properties to consider, the values of which are still not assigned. The resulting partial configuration can be represented as  $\vec{b}_O = (1, \cdot, \cdot)$ , where  $\cdot$  represents the unassigned values of remaining object properties. The set  $\mathcal{C}_{[\text{Lock\_Locked} \mapsto 1]}$  would then contain all possible configurations whose assignments are still compatible with  $\vec{b}_O = (1, \cdot, \cdot)$ : e.g.,  $\vec{b}'_O = (1, 1, 0)$  would be

in  $\mathcal{C}_{[Lock\_Locked \mapsto 1]}$ , while  $\vec{b}_O'' = (0, 1, 0)$  would be excluded from the set since the value of the first object property is set to zero (against  $[Lock\_Locked \mapsto 1]$ ).

We let  $\text{objRiskVal} = \sum_{a \in \langle o \rangle_A \cup \langle o \rangle_F} (\rho(a, \vec{b}_O)_{A,F} \times \text{Im}(a))$  represent the cumulative risk value on a specific OaR  $o$ , given events and actions in which it participates. Intuitively, we sum the risk values from each event/action in which  $o$  participates, resulting from the probability of each event/action times its impact factor. Given a set of configurations  $\mathcal{C}$ , we let  $\text{Val}_{\mathcal{C}}$  define semantics for layer 3 formulae:

$$\begin{aligned} \text{Val}_{\mathcal{C}}(\text{MostRisky}_A(o)) &= \arg\max_{a \in \langle o \rangle_A} \max_{\vec{b}_O \in \mathcal{C}} (\rho(a, \vec{b}_O)_{A,F} \times \text{Im}(a)); \\ \text{Val}_{\mathcal{C}}(\text{MostRisky}_F(o)) &= \arg\max_{a \in \langle o \rangle_F} \max_{\vec{b}_O \in \mathcal{C}} (\rho(a, \vec{b}_O)_{A,F} \times \text{Im}(a)); \\ \text{Val}_{\mathcal{C}}\left(\text{TotalRisk}(o)\right)_{\max} &= \max_{\vec{b}_O \in \mathcal{C}} \text{objRiskVal}; & \text{Val}_{\mathcal{C}}\left(\text{TotalRisk}(o)\right)_{\min} &= \min_{\vec{b}_O \in \mathcal{C}} \text{objRiskVal}; \\ \text{Val}_{\mathcal{C}}(\text{OptimalConf}(o)) &= \arg\min_{\vec{b}_O \in \mathcal{C}} \text{objRiskVal}; & \text{Val}_{\mathcal{C}}(\xi[op \mapsto bool]) &= \text{Val}_{\mathcal{C}_{\{op \mapsto bool\}}}(\xi) \end{aligned}$$

**Assumption 5:** Note that with semantics as given, the attacker can adapt its strategy to the node under consideration. E.g., when computing max total risk we assume the attacker can maximise the risk level for each individual node in the graph by choosing the best BASes at each iteration. We then sum risk levels derived from each of these single-node worst-case scenarios.

## 7 Related work

This paper directly relates to approaches that seek to combine ATs and FTs and increase their expressive capabilities. In this sense, numerous works attempt combinations of FTs and ATs into joint safety-security models: these are collected in a recent survey on model-based formalisms for safety-security risk assessment [27]. Of the 14 selected formalisms in [27], 7 combine or extend FTs and ATs: Attack-Fault Trees [2], Component Fault Trees [22], Extended Fault Trees – also known as Fault Trees with Attacks [11], Boolean driven Markov processes (BDMPs) [5], Attack Tree Bow Ties [1], Failure-Attack-Countermeasure Graphs [32], and State/Event Fault Trees (SEFTs) [29]. Of these formalisms, only BDMPs and SEFTs explicitly integrate properties of objects by joining FTs and Petri nets, expressing that certain disruptions can only happen in certain states. However, both BDMPs and SEFTs do not explicitly address how 1. the *part-hood* relation between objects and 2. the *participation* relation between objects and events/actions can influence the propagation of disruptions and the computations of risk levels. Furthermore, they do not allow for aggregation of risk levels on a given object, nor do they allow to compute an optimal configuration of states to minimize risk. Lastly, [20] presents Object-Oriented Fault Trees (OFTs), where each FT node is described by an object with instance variables containing information such as the node's parents, children and type. Despite the name, OFTs do not account for objects participating in different events represented by FT nodes, nor can they account for risk aggregation on objects, given safety- and security-related events/actions.



## 8 Discussion

To validate our approach, we intend to perform requirement elicitation from users: we then plan to translate these elicited requirements into concrete queries, modeled after those in [Sec. 5](#) that serve for now a simple exemplification purpose. The effectiveness of our approach will be evaluated on the basis of the types of queries we are able to capture. Additionally, we aim to conduct user studies via the development of a mock-up implementation to further evaluate the approach hereby presented. Ideally, we envision users to be risk analysts, as our approach is a natural extension of models that they might already be familiar with. Finally, as far as scalability is concerned, we expect to formulate novel symbolic model-checking algorithms that encode presented computational semantics in Binary Decision Diagrams (BDDs) [6]: BDDs are promising as they have proven to be computationally successful and efficient already in the classical setting of FT and AT analysis [4].

It is important to notice that in the case of *operational* assumptions – stricter than the absolute necessary – we take care to never be incoherent with COVER: i.e., we could lift or weaken these assumptions in future work, while remaining still grounded in the COVER ontology. E.g., one could weaken **assumption 3** to account for the principle of mereological expansion or **assumption 4** – fundamental for risk computation – when considering a different ordering between failures and attacks, or again **assumption 5** to model different types of attackers. These diverse possibilities increase flexibility in modelling disruptive situations while remaining grounded in COVER.

## 9 Conclusion and future work

We presented WATCHDOG, an ontology-aware framework for object-oriented risk assessment that exploits both the strengths of model-based formal methods and ontologies: by combining ontologies with probabilistic risk quantification models, we enriched the expressive power of FTs and ATs and presented a more expressive ontology-aware model (DOGs), logic (DOGLog) and a query language (DOGLang). We chose COVER for its domain-independent nature and its foundation in a comprehensive analysis of existing work on risk ontologies. However, our approach remains flexible and does not preclude the integration of knowledge from other ontologies. Our research opens up interesting directions for future work. First, one could allow a more nuanced notion of propagation or of parthood, considering the parthood relationship of OaRs and mereology. Furthermore, one could enrich DOGs by introducing new concepts from (the COVER) ontology, e.g., the notion of *goal*, or by introducing multiple *risk assessors* viewpoints via multiple ATs and FTs components. Finally, one could consider the effect of weakening assumptions, as discussed in [Sec. 8](#).

## References

1. Abdo, H., Kaouk, M., Flaus, J.M., Masse, F.: A new approach that considers cyber security within industrial risk analysis using a cyber bow-tie analysis (2017)
2. Arnold, F., Guck, D., Kumar, R., Stoelinga, M.: Sequential and parallel attack tree modelling. In: Proc. SAFECOMP. pp. 291–299 (2015)
3. Aven, T., Renn, O., Rosa, E.A.: On the ontological status of the concept of risk. *Saf. Sci.* **49**(8), 1074–1079 (2011)
4. Basgöze, D., Volk, M., Katoen, J., Khan, S., Stoelinga, M.: BDDs Strike Back - Efficient Analysis of Static and Dynamic Fault Trees. In: (NFM). vol. 13260, pp. 713–732 (2022)
5. Bouissou, M., Bon, J.L.: A new formalism that combines advantages of fault-trees and markov models: Boolean logic driven markov processes. *RESS* (2003)
6. Brace, K.S., Rudell, R.L., Bryant, R.E.: Efficient implementation of a BDD package. In: 27th ACM/IEEE Design Automation Conference. pp. 40–45 (1990)
7. Clark, P., Harrison, P., Jenkins, T., Thompson, J.A., Wojcik, R.H., et al.: Acquiring and using world knowledge using a restricted subset of English. In: *Flairs conference*. pp. 506–511 (2005)
8. Conrad, E., Titolo, L., Giannakopoulou, D., Pressburger, T., Dutle, A.: A compositional proof framework for FRETish requirements. In: *CCP*. pp. 68–81 (2022)
9. Crapo, A., Moitra, A., McMillan, C., Russell, D.: Requirements capture and analysis in ASSERT (TM). In: *RE*. pp. 283–291. *IEEE* (2017)
10. Dutuit, Y., Rauzy, A.: A linear-time algorithm to find modules of fault trees. *IEEE Transactions on Reliability* **45**(3), 422–425 (1996)
11. Fovino, I.N., Masera, M., De Cian, A.: Integrating cyber attacks within fault trees. *Reliability Engineering & System Safety* **94**(9), 1394–1402 (2009)
12. Fumagalli, M., Engelberg, G., Sales, T.P., Oliveira, Í., Klein, D., Soffer, P., Baratella, R., Guizzardi, G.: On the semantics of risk propagation. In: *RCIS* (2023)
13. Guarino, N.: On the semantics of ongoing and future occurrence identifiers. In: *ER 2017*. vol. 10650, pp. 477–490. *Springer* (2017)
14. Guarino, N., Baratella, R., Guizzardi, G.: Events, their names, and their synchronic structure. *Applied Ontology* **17**(2), 249–283 (2022)
15. Guizzardi, G., Wagner, G., de Almeida Falbo, R., Guizzardi, R.S., Almeida, J.P.A.: Towards ontological foundations for the conceptual modeling of events. In: *ER*. pp. 327–341. *Springer* (2013)
16. Guizzardi, G., et al.: UFO: Unified foundational ontology. *Appl. Ont.* **17**(1) (2022)
17. International Standardization Organization: ISO/DIS 26262: Road vehicles, functional safety. <https://www.iso.org/standard/68383.html> (2018)
18. ISO: Risk Management - Vocabulary, ISO Guide 73:2009 (2009)
19. Isograph: AttackTree. [www.isograph.com/software/attacktree/](http://www.isograph.com/software/attacktree/) (Acc Mar 2023)
20. Iverson, D.L., Patterson-Hine, F.: A diagnosis system using object-oriented fault tree models. *Proc. Artificial Intelligence for Space Applications* pp. 341–9 (1990)
21. Jürjens, J.: UMLsec: Extending UML for secure systems development. In: *UML~2002 — The Unified Modeling Language*. vol. 2460, pp. 412–425 (2002)
22. Kaiser, B., Liggesmeyer, P., Mäkel, O.: A new component concept for fault trees. In: *SCS*. pp. 37–46. *Citeseer* (2003)
23. Kriaa, S., Bouissou, M., Colin, F., Halgand, Y., Pietre-Cambacedes, L.: Safety and security interactions modeling using the BDMP formalism: case study of a pipeline. In: *SAFECOMP*. pp. 326–341. *Springer* (2014)

24. Lopuhaä-Zwakenberg, M., Budde, C.E., Stoelinga, M.: Efficient and generic algorithms for quantitative attack tree analysis. *IEEE TDSC* (2022)
25. Nicoletti, S.M., Lopuhaä-Zwakenberg, M., Hahn, E.M., Stoelinga, M.: Pfl: A probabilistic logic for fault trees. In: *FM 2023*. pp. 199–221 (2023)
26. Nicoletti, S.M., Lopuhaä-Zwakenberg, M., Hahn, E.M., Stoelinga, M.: ATM: A Logic for Quantitative Security Properties on Attack Trees. In: *SEFM* (2023)
27. Nicoletti, S.M., Peppelman, M., Kolb, C., Stoelinga, M.: Model-based joint analysis of safety and security: Survey and identification of gaps. *Comp. Sci. Rev.* **50** (2023)
28. Pease, A., Murray, W.: An english to logic translator for ontology-based knowledge representation languages. In: *NLP-KE*. pp. 777–783. *IEEE* (2003)
29. Roth, M., Liggesmeyer, P.: Modeling and Analysis of Safety-Critical Cyber Physical Systems using State/Event Fault Trees. In: *SAFECOMP* (2013)
30. Roudier, Y., Apvrille, L.: SysML-Sec: A model driven approach for designing safe and secure systems. In: *MODELSWARD*. pp. 655–664. *IEEE* (2015)
31. Ruijters, E., Stoelinga, M.: Fault Tree Analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Comp. Sci. Rev.* **15–16**, 29–62 (2015)
32. Sabaliauskaite, G., Mathur, A.P.: Aligning cyber-physical system safety and security. In: *Complex Systems Design & Management Asia*, pp. 41–53. Springer (2015)
33. Sales, T.P., et al.: The common ontology of value and risk. In: *ER* (2018)
34. Schneier, B.: Attack trees. *Dr. Dobbs’s journal* **24**(12), 21–29 (1999)
35. Stoelinga, M., Kolb, C., Nicoletti, S.M., Budde, C.E., Hahn, E.M.: The marriage between safety and cybersecurity: Still practicing. In: *SPIN*. pp. 3–21 (2021)
36. Sun, M., Mohan, S., Sha, L., Gunter, C.: Addressing safety and security contradictions in cyber-physical systems. In: *CPSSW*. Citeseer (2009)
37. White, C., Schwitter, R.: An update on PENG light. In: *ALTA*. pp. 80–88 (2009)
38. Zio, E.: The future of risk assessment. *Reliability Engineering & System Safety* **177**, 176–190 (2018)