



An Ontology of Security from a Risk Treatment Perspective

Ítalo Oliveira¹(✉), Tiago Prince Sales¹, Riccardo Baratella¹,
Mattia Fumagalli¹, and Giancarlo Guizzardi^{1,2}

¹ Conceptual and Cognitive Modeling Research Group (CORE),
Free University of Bozen-Bolzano, Bolzano, Italy
{idasilvaoliveira,tprincesales,rbaratella,mfumagalli,gguizzardi}@unibz.it

² Services and Cybersecurity Group, University of Twente,
Enschede, The Netherlands

Abstract. In Risk Management, security issues arise from complex relations among objects and agents, their capabilities and vulnerabilities, the events they are involved in, and the value and risk they ensue to the stakeholders at hand. Further, there are patterns involving these relations that crosscut many domains, ranging from information security to public safety. Understanding and forming a shared conceptualization and vocabulary about these notions and their relations is fundamental for modeling the corresponding scenarios, so that proper security countermeasures can be devised. Ontologies are instruments developed to address these conceptual clarification and terminological systematization issues. Over the years, several ontologies have been proposed in Risk Management and Security Engineering. However, as shown in recent literature, they fall short in many respects, including generality and expressivity - the latter impacting on their interoperability with related models. We propose a *Reference Ontology for Security Engineering (ROSE)* from a Risk Treatment perspective. Our proposal leverages on two existing Reference Ontologies: the *Common Ontology of Value and Risk* and a *Reference Ontology of Prevention*, both of which are grounded on the *Unified Foundational Ontology (UFO)*. ROSE is employed for modeling and analysing some cases, in particular providing clarification to the semantically overloaded notion of Security Mechanism.

Keywords: Risk Management · Security Engineering · Ontology

1 Introduction

In Risk Management, security issues arise from complex relations among objects and agents, their capabilities and vulnerabilities, the events they participate in, and the value and risk they ensue to the stakeholders at hand. Moreover, there

Work supported by Accenture Israel Cybersecurity Labs.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
J. Ralyté et al. (Eds.): ER 2022, LNCS 13607, pp. 365–379, 2022.
https://doi.org/10.1007/978-3-031-17995-2_26

are patterns involving these relations that crosscut many domains, including aviation, information systems, chemical industry, public safety, and national defence [16]. Understanding the details and having a shared conceptualization and vocabulary about those notions and their relations is fundamental for modeling and analysing the corresponding scenarios, so that proper security countermeasures can be devised. Once this conceptualization task is done in a proper way, it is possible to model and reason about actual and possible scenarios to assess and counter the risks through security mechanisms.

Models representing risk and security scenarios play an important role in the understanding, analysis, communication, and training in Risk Management. They provide guidance regarding what questions should be asked, and the type of data that should be collected; they establish relations between pieces of information and help giving meaning to data; they define the ways risks can be treated; they provide a shared conceptual framework among stakeholders which support communication and training [19]. Ontologies are instruments developed in many domains to address the tasks related to conceptual clarification and terminological systematization. Indeed, the need of a general security ontology was already noticed in [9] as a way of rigorously organizing the knowledge about security of information systems, helping to report incidents more effectively, share data and information across organizations. Then, several ontologies have been proposed in Risk Management and Security Engineering to offer support for many conceptual problems and applications. For example, risk and security assessment [23]; data integration and interoperability [7]; simulation of threats to corporate assets [11]. In parallel, domain-specific modeling languages, such as CORAS [20], Bowtie Diagrams [27], and the risk and security overlay of the ArchiMate language [3], implicitly assume an ontology of risk and security in their modeling constructs. An adequate reference ontology of this domain would be able to analyse, (re)design, and integrate languages like these, improving their modeling capabilities, in way analogous to how the *Common Ontology of Value and Risk* (COVER) [24] has been used to redesign Archimate w.r.t. risk and value modeling (e.g., [25]).

Existing proposals for conceptualizing risk and security - counting current security core ontologies and the metamodels of domain-specific modeling languages - fall short in many respects, including generality (e.g., they tend to suffer from premature domain optimization) and expressivity (e.g., they tend to be represented through ontologically neutral modeling languages, missing ontological distinctions) - the latter impacting on their interoperability with related models. Often, these problems come from the fact that these models are designed as lightweight ontologies (i.e., focused on computational aspects) as opposed to Reference ontologies (i.e., focused on ontological precision and conceptual adequacy). For example, although CORAS language, Bowtie diagrams and ArchiMate language show an appropriate degree of generality for representing different scenarios, they are informal languages which, in ontological terms, conflate the *object*, its *capability*, and the associated *event* and *situation* with regard to security mechanisms. On the other hand, security core ontologies presented in

computational logic languages, such as OWL, are often narrow by having specific applications in mind, missing at least the desirable generality. In fact, these core ontologies of security even fall short w.r.t. the *FAIR principles*, i.e., basic management standards for scientific artifacts (as shown by [22]).

To address these limitations, we employ an Ontology-Driven Conceptual Modeling approach [28] to propose a *Reference Ontology for Security Engineering (ROSE)* from a Risk Management perspective. The primary purpose of ROSE is to support activities related to ISO 31000 so-called *Risk Treatment* process [16]. Alternatively, one could refer to this as *security engineering of cybersocial systems*, because of the nature and pervasiveness of the problem of devising mechanisms for controlling and preventing the risks in cyber-physical and social systems (e.g., woody gates, circuit breakers, antivirus software, lockdown norms). Our proposal leverages on two existing Reference Ontologies, namely, the *Common Ontology of Value and Risk* and a *Reference Ontology of Prevention*, both of which are grounded on the *Unified Foundational Ontology (UFO)*. ROSE is employed for modeling and analysing some cases, in particular providing clarification to the semantically overloaded notion of Security Mechanism.

In what follows, Sect. 2 presents the requirements we expect ROSE to fulfil. Section 3 presents the foundations on which our proposal is based, that is, the Unified Foundational Ontology, in particular its conceptualization of the phenomenon of prevention. Section 4 presents our main contribution, the reference ontology of security termed ROSE, reusing an extended and reinterpreted version of COVER. Section 5 shows how ROSE satisfies the proposed requirements. Section 6 discusses the main related works. Section 7 marks our final considerations.

2 Requirements for a Reference Ontology of Security

ISO 31000 [16] defines that the process of Risk Management involves communication and consultation about the risks, risk assessment, risk treatment, recording and reporting, and monitoring and review. In this view, the purpose of risk treatment is to select and implement options for addressing risk. Risk treatment involves an iterative process of: “(a) formulating and selecting risk treatment options; (b) planning and implementing risk treatment; (c) assessing the effectiveness of that treatment; (d) deciding whether the remaining risk is acceptable; (e) if not acceptable, taking further treatment” [16]. ISO 31000 states that selecting the most appropriate risk treatment option(s) involves balancing the potential benefits derived in relation to the achievement of the objectives against costs, effort or disadvantages of implementation. Risk treatment options include: “(a) avoiding the risk by deciding not to start or continue with the activity that gives rise to the risk; (b) taking or increasing the risk in order to pursue an opportunity; (c) removing the risk source; (d) changing the likelihood; (e) changing the consequences; (f) sharing the risk with another party or parties (including contracts and risk financing); and (g) retaining the risk by informed decision” [16]. ROSE can be seen as a way of *ontologically unpacking* [15] this notion of risk treatment, according to the risk treatment options, through an

ontological analysis based on the Unified Foundational Ontology (UFO) and its general-purpose conceptual modeling language OntoUML [12].

Taking these tasks into account, we distinguish three types of requirements that ROSE shall satisfy:

Analysis Requirements (AR): domain-specific capabilities associated to the tasks the ontology should help to realize:

1. Since security engineering requires risk assessment as a previous step in the risk management process, ROSE shall support the identification and assessment of risks;
2. ROSE shall support activities associated with security engineering in multiple domains.

Ontological Requirements (OR): domain-specific concepts and relations the ontology should have in order to realistically represent its domain of interest and thus support what it is intended to support:

1. ROSE shall support the task of representing the risk treatment options (a)–(g), which are directly connected to the AR2;
2. ROSE shall include both risk and security concepts, explaining explicitly how they interact with one another, including the ones mapped in [22] as the most common in security core ontologies: Vulnerability, Risk, Asset, Attacker, Threat, Control, Countermeasure, Stakeholder, Attack, Consequence;
3. ROSE shall be able to distinguish intentional and non-intentional threats, because this distinction impacts the risk treatment options.

Quality Requirements (QR): domain-independent characteristics the ontology is expected to possess, so it becomes a better artifact:

1. *Domain appropriateness* [12] - ROSE shall capture the relevant entities and relations of the domain through an ontological analysis;
2. *Generality* - Since security crosses multiple different areas, a security reference ontology should represent the most general concepts of the domain;
3. *FAIR principles* - ROSE should be Findable, Accessible, Interoperable and Reusable [17].

3 Ontological Foundations of Prevention

Our strategy to address the requirements described in Sect. 2 is to employ the Unified Foundational Ontology (UFO) - in particular, a module that covers the phenomenon of prevention. ROSE will be represented through the UFO-based modeling language OntoUML [12]. We are interested in UFO's conceptualization of prevention (presented in [4]), which involves multiple ways of stopping or forestalling certain types of events, because this sort of dynamics plays a fundamental

role in the domain of security. In Sect. 4, we build ROSE as an extension and reinterpretation of the Common Ontology of Value and Risk (COVER), applying the theory of prevention. COVER was chosen, because it includes several concepts and relations about value and risk that are crucial for a reference ontology of security. Indeed, COVER was used to evaluate and redesign ArchiMate language regarding to value and risk [25,26]. Additionally, COVER has been successfully applied for modeling different domains, such as trust [2], software anomalies [10], among others. However, COVER assumes specific future events are entities of its domain of discourse [24], an assumption that is inconsistent with UFO theory of events, which claims that particular events are immutable entities in the past or that are happening [14]. We adopt UFO assumption with the support of higher-order types to represent future events as types of events, and review some cardinality constraints in COVER.

UFO [12] distinguishes individuals and types: objects, dispositions, events, situations instantiating object types, disposition types, event types, and situation types, respectively. The same type-token distinction applies to relations. Types can be more or less saturated, depending on the presence of individual concepts in the type definition, provided the type can be instantiated by multiple individuals. A fully unsaturated type is defined only by general properties (e.g., the type “Physical Object” is defined by general properties such as spatial extension, weight, color). Individual concepts (e.g., the concept of “Facebook”), on the other hand, are fully saturated, i.e., they are instantiated by only one individual and always the same individual. A semi-saturated type is defined by general properties as well as individual concepts. For instance, the type “cyberattack against Facebook” includes the general type “cyberattack” and the individual concept “Facebook”, but it can be instantiated by multiple events.

According to UFO [4,14], events are manifestations of interacting objects’ dispositions, which are activated by certain situations. For example, the event of a lion attack on customers in a given zoo is the result of manifestations of capabilities of the lion and the vulnerabilities of customers, when these dispositions meet each other in the right situations that activate them. In this example, the vulnerabilities and the capabilities bear a *mutual activation partnership* relation among each other. If an event E_1 brings about a situation S that activates the dispositions that are manifested as event E_2 , then we say that S *triggers* E_2 , and that E_1 *directly-causes* E_2 ; if E_1 *directly-causes* E_2 , and E_2 *directly-causes* E_3 , then E_1 *causes* E_3 , where *causes* is a strict partial order relation [14]. So *causes* is the transitive closure of *directly-causes*. Furthermore, likelihood or probability can be ascribed to events of certain types, as described in [24] and incorporated in UFO theory of prevention [4] in the following way: TRIGGERING LIKELIHOOD inheres in a Situation Type, and it refers to how likely a Situation Type will trigger an Event Type once a situation of this type is brought about by an event; the CAUSAL LIKELIHOOD inheres in an Event Type, and it means the chances of an event causing, directly or indirectly, another one of a certain type.

This theory about the relations between situations, objects, their dispositions, and events implies that certain types of events can prevent other types of events due to some effect on dispositions, their partner dispositions, or the situation that could activate the dispositions. For instance, caging the lion prevents

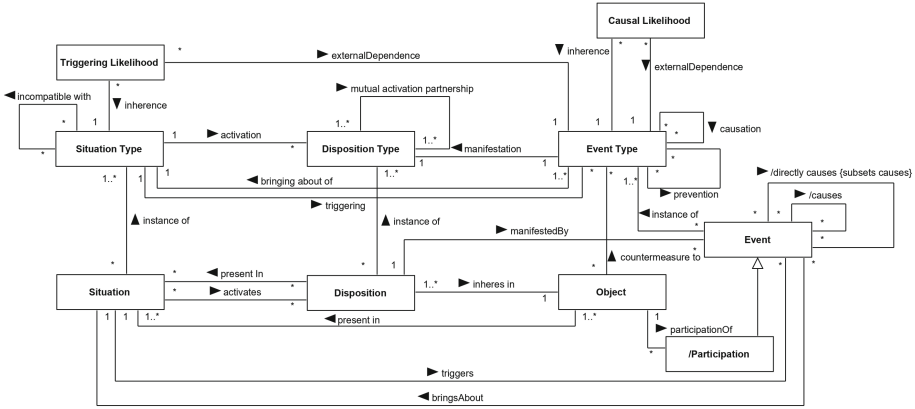


Fig. 1. Theory of prevention in UFO (adapted from [4]). Notice we added the relations *incompatible with*, *triggering*, and *causation* to the original diagram.

the formation of a situation in which capacities of the lion and the vulnerabilities of the customers could meet; sedating the lion would remove some of its threatening capabilities; in both cases, the (semi-saturated) type of event of the lion attack in the zoo would be prevented. In general, prevention of events of type E_T that are manifestations of dispositions of type D_T occurs when an event of type E'_T brings about a situation of a type S'_T that is incompatible with the situations required to activate instances of D_T [4]. This incompatibility means these situations cannot obtain concurrently, and, on the type-level, *incompatible*(S_T, S'_T) implies that there are no two instances of these two types that obtain in overlapping time intervals. These situation types are semi-saturated, that is, they must share some relevant dispositions or objects. E.g., a situation with updated software is compatible with a situation that contains a different outdated software, even if the two situations temporally overlap, though they would be incompatible if the referred software was the same individual in both situations.

Prevention can then be defined as a relation between two types of events: *prevention*(E_T, E'_T) implies that the occurrence of events of type E_T brings about situations that are incompatible with the conditions required for the occurrence of events of type E'_T . Again, these event and situation types are semi-saturated in the proper way, i.e., guaranteeing the presence (co-reference) of the same disposition and bearer. For example, it is the event of *Humidifying object x* that prevents an event of *Catching on Fire of object x*. Obviously, humidifying flammable objects, in general, does not prevent other flammable objects from catching on fire. Besides, notice an event (or a type of event) cannot hold a prevention relation with a specific event, which is always an immutable existing entity, but an event can prevent a type of event, therefore precluding the occurrence of instances of this type.

Given this definition, there is a sense in drawing a distinction between two types of *indirect prevention*. One way of producing indirect prevention is if an event e causes an event e' , and e' prevents events of type E_T , so we say e indirectly prevents E_T . Another way of producing indirect prevention is if an event e prevents events of type E_T , which is causally connected to E'_T , so we say e indirectly prevents E'_T . For example, an event *my car engine failure* causes the event *my car stops in the traffic*, which prevents the events of (semi-saturated) type *me attending the job interview*; if I had attended the job interview, *I would have gotten the job* - a type of event that is historically dependent on the events of type *me attending the job interview*. Indirect prevention plays an important role in security engineering, because Security Mechanisms (a) may produce a chain of events that eventually prevents directly the desired type of event or (b) may block a causal chain of undesired types of events.

UFO theory of prevention also defines a concept of *Countermeasures* [4]. In general, given a disposition d whose manifestations are of type E_T , countermeasures are designed interventions that endow a setting containing d with other dispositions $\{d_1, \dots, d_n\}$, whose manifestations prevent any instance of E_T . More specifically, *Countermeasure Mechanisms* are designed such that: they contain dispositions of type D_T , and given the situations of type S_T that would trigger events that would (directly or indirectly) cause instances of E_T , the instances of S_T instead activate the instances D_T whose associated event type prevent E_T . For example, a circuit breaker contains a disposition to close the circuit in a situation where there is a current above a certain threshold. The manifestation of that disposition of the circuit breaker thus prevents the event of an overcurrent.

This analysis makes explicit several ways in which countermeasures can be designed [4]: (i) removing the disposition d whose manifestation we want to avoid (this can be done by removing the object with that disposition from the setting at hand); (ii) removing from the scene required activation partners (e.g., produce a vacuum to prevent fires); (iii) including in that setting a disposition that is incompatible with a mutual activation partner (e.g., humidifying a flammable object, removing dryness as a required property); (iv) designing countermeasure mechanisms surrounding the bearer of d , which have the capacity of preventing the manifestation of d . The theory of prevention is summarized in Fig. 1.

4 A Reference Ontology for Security Engineering (ROSE)

Our approach is to understand the domain of security as the *intersection* between the domain of value and risk, understood under the terms of COVER [24], and the theory of prevention [4]. In this sense, security mechanism creates value protecting certain goals from risk events. In COVER, Value is a relational property that emerges from the relations between capacities of certain objects and the goals of an agent. The manifestations of these capacities are events that bring about a situation that impacts or satisfies the goal of a given agent - the goal is simply the propositional content of an intention [13]. Risk is the anti-value: risk events are the manifestations of capacities, vulnerabilities and, sometimes,

intentions that inhere in an agent; these events bring about a situation that hurts the goal of a given agent. Like value, security is a relational property that emerges from the relations between capabilities of objects and goals of an agent; the manifestations of these capabilities bring about a situation that impacts the goal of an agent in a very specific way: preventing risk events. In what follows we develop this conceptualization, firstly by extending COVER, then by presenting an ontology of security.

4.1 Extending the Common Ontology of Value and Risk

In COVER, RISK EVENT is the result of the manifestations of THREAT CAPABILITY of THREAT OBJECT and VULNERABILITY of OBJECT AT RISK or of RISK ENABLER. A THREAT EVENT is one with the potential of causing a LOSS EVENT, which brings about a LOSS SITUATION that hurts an INTENTION of an AGENT called RISK SUBJECT [24].

The assumption that a THREAT EVENT can be intentional is implicit in COVER, so we make it explicit specializing it through the class ATTACK, an ACTION caused by an INTENTION of an AGENT called ATTACKER, which specializes THREAT OBJECT. Traditionally, the presence or not of intention in a THREAT EVENT is raised to set the difference between security and safety, respectively [5], though in both cases the goal is the prevention of the LOSS EVENT.

An important addition to COVER is the understanding that THREAT CAPABILITY, VULNERABILITY and, sometimes, INTENTION are dispositions associated to types whose instances maintain a mutual activation partnership to each other: a THREAT OBJECT can only manifest its THREAT CAPABILITY if a VULNERABILITY can be exploited; if the THREAT OBJECT creates an ATTACK, then the INTENTION is also required. Analogously, a VULNERABILITY is only manifested in the presence of a THREAT CAPABILITY. This *generic dependence* relation among these entities determines some ways by which Security Measures can work: the removal of any of them from the situation that could activate them all together implies the prevention of the associated RISK EVENT.

We need to mention briefly the notions of VALUE ASCRIPTION and RISK ASSESSMENT in COVER, because they are able to represent the quantification of value and risk [24]. In a nutshell, an AGENT, called VALUE ASSESSOR or RISK ASSESSOR, evaluates her value and risk experience, considering the satisfaction or dissatisfaction of her goals by the manifestation of dispositions of certain objects. This judgment is then a reified relational entity to which a *quality* can be assigned - for example, like in the severity scale of risk matrix with discrete or continuous values (e.g., $\langle Low, Medium, High \rangle$).

Our extension of COVER is represented in OntoUML language in Fig. 2. The colors used signal the corresponding UFO categories: object types are represented in pink, intrinsic aspect types in light blue, situation types in orange, event types in yellow, higher-order types in a darker blue.

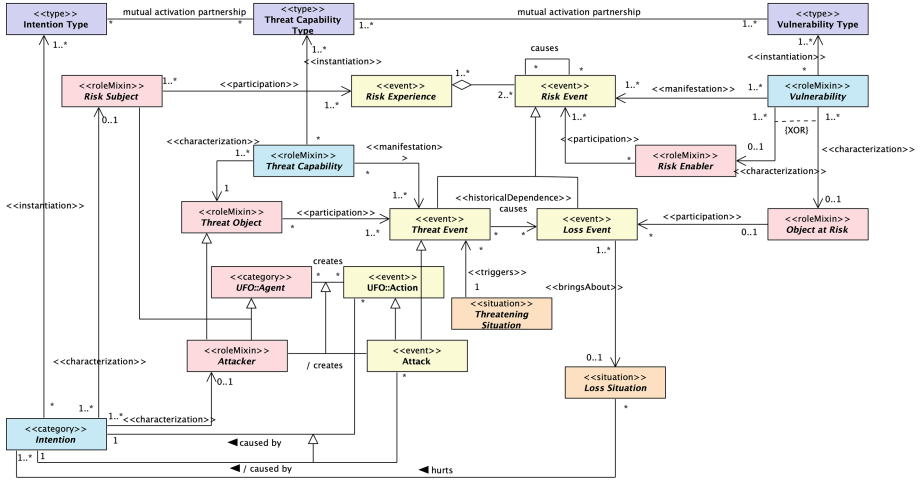


Fig. 2. COVER extension concerning risk concepts and relations.

4.2 Unpacking the Notion of Security Mechanism

A SECURITY MECHANISM is always designed by an AGENT called the SECURITY DESIGNER to be a *countermeasure* to events of certain type (RISK EVENT TYPE). The AGENT creating a SECURITY MECHANISM is not necessarily the one who is protected by its proper functioning, i.e., the PROTECTED SUBJECT. Both agents, nonetheless, have INTENTIONS that are positively impacted by this proper functioning. For example, the government designs policies for public safety, the functioning of such policies satisfies some goal the government had when it designed them, but also satisfies the goal of people who want to be safe. Sometimes, the PROTECTED SUBJECT is the same AGENT as the SECURITY DESIGNER, like when a person builds a wall for their own house.

An INTENTION can be generic or specific, according to how specific the situation that satisfies it is. For example, in aerospace domain some goals related to the costs of the mission are generic, because they can be satisfied by more funding or an assurance; even goals related to replaceable engineering parts can be satisfied by other parts of the same type. However, the completion of the mission is a specific goal that can only be satisfied by a specific situation. This distinction is important, because certain security mechanisms only work for generic goals. For instance, a space company that transfers some of its risks to an insurance company can be protected from financial loss, but not from the losses caused by the explosion of a space shuttle. Ultimately, GENERIC INTENTION can only be impacted by a setting with generic VALUE OBJECTS (money, for example), but the SPECIFIC INTENTION may be satisfied by a specific setting with generic VALUE OBJECTS (say, the need for money under a deadline of bankruptcy).

A SECURITY MECHANISM is an object, which may be a simple physical object like a wall, a high-tech air defense system like the Israeli Iron Dome, an AGENT

like a policeman, a social entity like a security standard or anti-COVID-19 rules, that bears dispositions called CONTROL CAPABILITY. The manifestation of this kind of disposition is a PROTECTION EVENT, specialized in CONTROL CHAIN EVENT and CONTROL EVENT, where the former can cause the latter. The CONTROL EVENT is of a type (CONTROL EVENT TYPE) that prevents, directly or indirectly, events of certain type (RISK EVENT TYPE). This is so because the control events bring about a CONTROLLED SITUATION, which is of a type that is *incompatible with* the situations of the type that triggers risk events of certain types. Since risk events are specialized in THREAT EVENT and LOSS EVENT, the CONTROLLED SITUATION TYPE is incompatible with the THREATENING SITUATION TYPE or with the LOSS TRIGGERING SITUATION TYPE. Figure 3 shows this ontological unpacking of the notion of Security Mechanism.

Consider an antivirus software (SECURITY MECHANISM) in Anna's computer (Anna is a PROTECTED SUBJECT, but also a RISK SUBJECT). It was designed by a software company (SECURITY DESIGNER) that has its own interest in seeing the antivirus capability (CONTROL CAPABILITY), under the right settings (PROTECTION TRIGGER), working properly (manifesting the PROTECTION EVENT). Under the right settings (PROTECTION TRIGGER), the antivirus searches for malware (the very search can be considered a CONTROL CHAIN EVENT of the causal chain, while the malware as a software is a THREAT OBJECT). Suppose Anna's computer is infected by a malware, which was a THREAT EVENT in the process of causing a LOSS EVENT (say, erasing Anna's files in her computer, where her files are the OBJECT AT RISK). This event of infection (an ATTACK) was only possible due to the conjunction of malicious INTENTION of someone (an ATTACKER), the THREAT CAPABILITY of this person, and the VULNERABILITY of Anna or her computer (RISK ENABLER). However, before the manifestation of the LOSS EVENT, as the antivirus software is running, an event in the control chain causes a CONTROL EVENT of a type that is incompatible with the LOSS TRIGGERING SITUATION TYPE (say, the situations that activate the execution of malware's code to delete Anna's files), therefore preventing the LOSS EVENT of certain type (the loss of Anna's files) that would have hurt Anna's goals. Instead, a CONTROLLED SITUATION (say, Anna's computer free from the referred malware) became a fact brought about by the CONTROL EVENT (say, interrupting the malware running process and deleting it), impacting positively Anna's goals.

Notice that both kinds of indirect prevention play an important role in security: (1) when a CONTROL CHAIN EVENT indirectly prevents a RISK EVENT TYPE through the causation of a CONTROL EVENT, which prevents directly a RISK EVENT TYPE; and (2) when the CONTROL EVENT prevents indirectly the LOSS EVENT TYPE that is causally connected to the directly prevented THREAT EVENT TYPE. In the next section we show how ROSE, which includes the extended ontology of value and risk from COVER as well as the ontology of security represented in Fig. 3, satisfies the requirements proposed in Sect. 2.

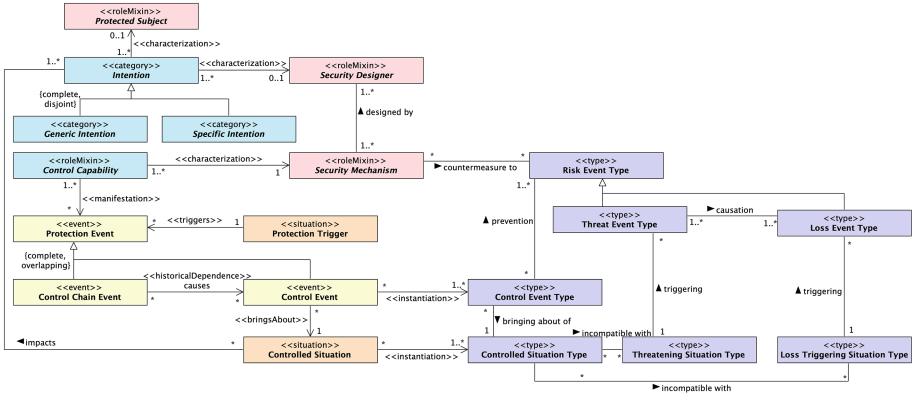


Fig. 3. Unfolding security mechanism.

5 Evaluation

ROSE incorporates a modified extended version of COVER, hence capturing risk and value concepts that are necessary for supporting the activity of risk identification and assessment. Indeed, by the notions of TRIGGERING LIKELIHOOD and CAUSAL LIKELIHOOD ascribed to types of situations and events, ROSE is able to support probabilistic assessment of value, risk, and security happenings. By the VALUE ASCRIPTION and RISK ASSESSMENT inherited from COVER, ROSE is able to support comparison of choices involving values, risks, and security, so supporting decision-making process, since in Risk Management the chances as well as the impact of risky and valuable options should be considered. This machinery allows for the representation of risk treatment options (a) and (b) of ISO 31000: the first case regards to the case that, from the point of view of the experience of the same AGENT that is VALUE ASSESSOR and RISK ASSESSOR, the risk is assessed as higher than the expected ascribed value; in this scenario, considering the chances of the respective events, the AGENT may choose not to start or to continue the activity; the second case is the opposite, when all things considered, the possible success of the endeavor is assessed by the AGENT as more valuable than its associated risks; in this scenario, the AGENT may choose to pursue an opportunity, despite of the risks.

ROSE shows the ambiguity of the risk treatment option (c) “removing the risk source” of ISO 31000, since there are multiple interacting entities that can be considered the “risk source”: instances of THREAT OBJECT, OBJECT AT RISK, RISK ENABLER, THREAT CAPABILITY, INTENTION, and VULNERABILITY. It is possible to remove the THREAT CAPABILITY without removing the THREAT OBJECT, though removing the latter implies the removal of the former due to the existential dependence of dispositions on their bearers. For example, a caged lion in a zoo is in such a situation, brought about by a caging event, such that its THREAT CAPABILITY and the VULNERABILITY of the visitors cannot be present in the same situation, though both dispositions remain untouched. However, if

the lion escapes and it is sedated by a dart gun shot, the lion, while unconscious, loses its **THREAT CAPABILITY**.

ROSE also shows the risk treatment options (c), (d) and (e) of ISO 31000 are interconnected. Since the **PROTECTION EVENT** is an instance of **EVENT TYPE** that has its associated **TRIGGERING LIKELIHOOD** and **CAUSAL LIKELIHOOD**, the effect of prevention happens with a given likelihood. This means the chances of risks events of a certain type happening are different before and after the introduction of the **SECURITY MECHANISM**. The “consequences” of risk treatment option (e) is simply the loss events of certain types, but a **LOSS EVENT** can be the **THREAT EVENT** that causes another **LOSS EVENT** - for example, a fire in the university office is a **LOSS EVENT** for the university, but it is a **THREAT EVENT** that can potentially harm the lives of employees.

Risk treatment option (f) of ISO 31000, “sharing the risk with another party or parties (including contracts and risk financing)”, was already described as a **SECURITY MECHANISM** that is only applicable to protect **GENERIC INTENTION** by the dispositions of interchangeable **VALUE OBJECTS**. In this case, the money or the replaceable object may be lost, but, once the equivalent reposition takes place, the events of the type associated to the initial loss are prevented. So the initial loss is both a **LOSS EVENT** (the loss of money) and a **THREAT EVENT** for future losses (the consequences of that).

The last risk treatment option of ISO 31000 concerns to retaining the risk by informed decision, that is, the decision to be taken about *residual risks* [18], the risks left after the treatments. This option is a combination of the previous ones: it says that, once options (c), (d), (e), and (f) are implemented, we return to the options (a) and (b) in an iterative decision process, as described in the standard [16]. Again, ROSE can inform such decision-making process by representing scenarios involving value, risk, and security. Residual risks is known to be difficult to assess [18], but ROSE offers a precise picture about the scenario before and after the security mechanism implementation.

ROSE includes all concepts mapped in [22] as the most common in security core ontologies, as requested by OR2. Indeed, ROSE ontologically unpacks them and explains their interactions in details, also distinguishing between intentional and non-intentional threats, and how this matters for security. It is worth noting that ROSE allows for the representation of *redundancy* and multiple layers of protection in security engineering, as different security mechanisms can be designed to be countermeasures of the same type of **RISK EVENT**.

Concerning the quality requirements, ROSE has shown a rich set of ontological distinctions, thanks to the support of UFO and COVER, maintaining generality, which is noted, for example, by the fact the **SECURITY MECHANISM** can be an object of different kinds, including physical and social objects. ROSE reuses COVER with some modifications, showing a level of interoperability with a close domain (value and risk), which can be further exploited through connection to other UFO-based ontologies. As UFO and OntoUML are formally defined in First-Order Logic, having support for an OWL implementation¹, ROSE ben-

¹ See: <https://purl.org/nemo/gufo>.

efits from that, in terms of formality and capacities, since it is expressed in OntoUML, making the use of ROSE for supporting formal reasoning easier. Finally, to make ROSE findable and accessible [17], we provide it publicly in a repository with related information².

6 Related Work

The closest related works to ours are proposals of reference ontologies of security based on some foundational ontology. In a recent literature review [22], only four with this approach were found, while nearly all the 57 selected security core ontologies miss every FAIR principle. In [6] the authors propose an ISO-based information security domain ontology, represented in OWL and designed under the principles of the Basic Formal Ontology (BFO), to facilitate the management of standards-related documents and compliance in an Information Security Management System. [21] uses the upper ontology DOLCE to combine two other ontologies to support the activity of modeling security requirements, representing the proposed ontology in the Extended Backus-Naur Format. In [23], based on DOLCE, the authors continue a previous work presenting human factors in this ontology for cyber security operations. To the best of our knowledge, none of them is publicly available besides what can be found inside each corresponding paper. Moreover, they present a limited scope concerning security, given their respective specific aims.

There exist UFO-based ontologies addressing security or related concepts. In [29], the authors propose an ontology to support hazard identification using some UFO categories, though presenting the ontology in UML, instead of OntoUML. Specific security aspects are not addressed therein. The proposal of [1] is more related to ours: based on UFO, but represented in UML, a “Combined Security Ontology” (CSO) that could be aligned with other ontologies. In CSO, countermeasure is an ACTION and asset is a Kind. In ROSE, we take a different ontological interpretation of these notions. Regarding the former, an ACTION may be the manifestation of a CONTROL CAPABILITY of a SECURITY MECHANISM, countermeasures are OBJECTS, not necessarily AGENTS, e.g., a software firewall. Regarding the latter, the type Asset cannot be a Kind, because being an asset depends on the relations the object has with other entities: firstly, nothing is necessarily an asset, but only to the extent the thing’s dispositions, when manifested, satisfy someone’s goals; moreover, entities of different kinds can be assets. Thus, this notion would be better modelled as role mixin in OntoUML/UFO. So CSO does not seem to commit to UFO in its full extent. The Dysfunctional Analysis Ontology (DAO) [8] continues the Goal-Oriented Safety Management Ontology (GOSMO) and aims at providing a systematization of the goal-oriented dysfunctional analysis through a terminological clarification in order to prevent hazards. They are represented in UML and OWL, making the same (in our view, mistaken) choice of interpreting SAFETY MEASURES as an ACTIONS.

² See: <https://github.com/unibz-core/security-ontology>.

7 Final Considerations

We have presented an ontological analysis of security mechanism, making explicit the relations among objects and agents, their capabilities and vulnerabilities, the events they participate and that affect them, and the value and risk they ensue to the stakeholders at hand. The result of this analysis was a concrete artifact called *Reference Ontology of Security Engineering* (ROSE), filling a gap left by the Common Ontology of Value and Risk (COVER) that lacked security-related concepts. With the support of the theory of prevention from the Unified Foundational Ontology, our ontology shows the different general ways by which a security mechanism works. In the future, we intend to combine ROSE with other UFO-based ontologies, in particular to address legal aspects, and to employ ROSE in the process of evaluating and (re)designing ArchiMate language.

References

1. Adach, M., et al.: A combined security ontology based on the unified foundational ontology. In: International Conference on Semantic Computing, pp. 187–194 (2022)
2. Amaral, G., Sales, T.P., Guizzardi, G., Porello, D.: Towards a reference ontology of trust. In: Panetto, H., et al. (eds.) OTM 2019. LNCS, vol. 11877, pp. 3–21. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33246-4_1
3. Band, I., Engelsman, W., Feltus, C., Paredes, S.G., Diligens, D.: Modeling enterprise risk management and security with the ArchiMate language (2015)
4. Baratella, R., Fumagalli, M., Oliveira, Í., Guizzardi, G.: Understanding and modeling prevention. In: Guizzardi, R., Ralyte, J., Franch, X. (eds.) Research Challenges in Information Science. RCIS 2022. LNBIP, vol. 446, pp. 389–405. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-05760-1_23
5. van den Berg, B., Hutten, P., Prins, R.: Security and safety: an integrative perspective. In: Jacobs, G., Suojanen, I., Horton, K.E., Bayerl, P.S. (eds.) International Security Management. ASTSA, pp. 13–27. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-42523-4_2
6. Casola, V., et al.: A first step towards an ISO-based information security domain ontology. In: International Conference on Enabling Technologies, pp. 334–339 (2019)
7. Chen, B., et al.: Research on ontology-based network security knowledge map. In: International Conference on Cloud Computing, Big Data and Blockchain, pp. 1–7 (2018)
8. Debbech, S., et al.: An ontological approach to support dysfunctional analysis for railway systems design. J. Univers. Comput. Sci. **26**(5), 549–582 (2020)
9. Donner, M.: Toward a security ontology. IEEE Secur. Priv. **1**(03), 6–7 (2003)
10. Duarte, B.B., de Almeida Falbo, R., Guizzardi, G., Guizzardi, R., Souza, V.E.S.: An ontological analysis of software system anomalies and their associated risks. Data Knowl. Eng. **134**, 101892 (2021)
11. Ekelhart, A., Fenz, S., Klemen, M.D., Weippl, E.R.: Security ontology: simulating threats to corporate assets. In: Bagchi, A., Atluri, V. (eds.) ICISS 2006. LNCS, vol. 4332, pp. 249–259. Springer, Heidelberg (2006). https://doi.org/10.1007/11961635_17
12. Guizzardi, G.: Ontological foundations for structural conceptual models (2005)

13. Guizzardi, G., et al.: Grounding software domain ontologies in the unified foundational ontology (UFO): The case of the ODE software process ontology. In: Ibero-American Conference on Software Engineering, pp. 127–140 (2008)
14. Guizzardi, G., Wagner, G., de Almeida Falbo, R., Guizzardi, R.S.S., Almeida, J.P.A.: Towards ontological foundations for the conceptual modeling of events. In: Ng, W., Storey, V.C., Trujillo, J.C. (eds.) ER 2013. LNCS, vol. 8217, pp. 327–341. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41924-9_27
15. Guizzardi, G., Bernasconi, A., Pastor, O., Storey, V.C.: Ontological unpacking as explanation: the case of the viral conceptual model. In: Ghose, A., Horkoff, J., Silva Souza, V.E., Parsons, J., Evermann, J. (eds.) ER 2021. LNCS, vol. 13011, pp. 356–366. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-89022-3_28
16. ISO: ISO 31000:2018 - Risk management - Guidelines (2018)
17. Jacobsen, A., et al.: FAIR principles: interpretations and implementation considerations. *Data Intell.* **2**(1–2), 10–29 (2020)
18. Katsikas, S.K.: Risk management. In: Vacca, J.R. (ed.) *Computer and Information Security Handbook*, pp. 507–527. Morgan Kaufmann, 3 edn. (2013)
19. Kjellén, U.: *Prevention of Accidents Through Experience Feedback*. CRC Press, Boca Raton (2000)
20. Lund, M.S., Solhaug, B., Stølen, K.: *Model-Driven Risk Analysis: The Coras Approach*. Springer, Heidelberg (2010)
21. Massacci, F., Mylopoulos, J., Paci, F., Tun, T.T., Yu, Y.: An extended ontology for security requirements. In: Salinesi, C., Pastor, O. (eds.) CAiSE 2011. LNBIP, vol. 83, pp. 622–636. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22056-2_64
22. Oliveira, Í., et al.: How FAIR are security core ontologies? A systematic mapping study. In: *Research Challenges in Information Science*, pp. 107–123 (2021)
23. Oltramari, A., et al.: Towards a human factors ontology for cyber security. *Semant. Technol. Intell. Def. Secur.* **2015**, 26–33 (2015)
24. Sales, T.P., Baião, F., Guizzardi, G., Almeida, J.P.A., Guarino, N., Mylopoulos, J.: The common ontology of value and risk. In: Trujillo, J.C., et al. (eds.) ER 2018. LNCS, vol. 11157, pp. 121–135. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00847-5_11
25. Sales, T.P., et al.: Ontological analysis and redesign of risk modeling in ArchiMate. In: *International Enterprise Distributed Object Computing Conference*, pp. 154–163 (2018)
26. Sales, T.P., Roelens, B., Poels, G., Guizzardi, G., Guarino, N., Mylopoulos, J.: A pattern language for value modeling in ArchiMate. In: Giorgini, P., Weber, B. (eds.) CAiSE 2019. LNCS, vol. 11483, pp. 230–245. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21290-2_15
27. Saud, Y.E., Israni, K., Goddard, J.: Bow-tie diagrams in downstream hazard identification and risk assessment. *Process Saf. Prog.* **33**(1), 26–35 (2014)
28. Verdonck, M., et al.: Ontology-driven conceptual modeling: a systematic literature mapping and review. *Appl. Ontol.* **10**(3–4), 197–227 (2015)
29. Zhou, J., et al.: An ontological approach to identify the causes of hazards for safety-critical systems. In: *System Reliability and Safety*, pp. 405–413 (2017)