Friend, Foe, or Target? Domain Models as Risk Deterrents, Risk Sources, and Assets at Risk

Isadora Valle[1(✉)], Tiago Prince Sales[1], Eduardo Guerra[2], Ítalo Oliveira[1], Renata Guizzardi[1], Luiz Olavo Bonino da Silva Santos[1], Henderik Proper[3], and Giancarlo Guizzardi[1]

[1] University of Twente, Enschede, The Netherlands
{i.vallesousa,t.princesales,i.j.dasilvaoliveira,r.guizzardi,
l.o.boninodasilvasantos,g.guizzardi}@utwente.nl
[2] Faculty of Engineering, Free University of Bozen-Bolzano, Bolzano, Italy
eduardo.guerra@unibz.it
[3] TU Wien, Vienna, Austria
henderik.proper@tuwien.ac.at

Abstract. Modelers and organizations often struggle to assess the benefits and drawbacks of modeling activities. This paper proposes addressing this challenge through a risk-oriented lens, leveraging the Common Ontology of ValuE and Risk (COVER) and the Reference Ontology for Security Engineering (ROSE). The proposal focuses on identifying assets at risk throughout the modeling process to clarify: when models mitigate risks and contribute to cost savings (models as risk deterrents), when models introduce risk to other assets (models as risk sources), or when they are vulnerable to risk events themselves (models as assets at risk), potentially generating additional costs. This perspective enables modelers and organizations to evaluate the benefits and costs of modeling practices, aligning investments with organizational goals, while helping researchers identify gaps for enhancing modeling languages, methods, and tools. The proposal is evaluated by analyzing case studies from the literature and interviews with nine professionals and researchers.

Keywords: Risk Assessment · Modeling · Domain Models · Conceptual Models · Return on Modeling Efforts

1 Introduction

Previous studies have highlighted and demonstrated that modelers and organizations face challenges in assessing the benefits and drawbacks of modeling activities. In [14], the authors introduced the concept of Return on Modeling Efforts (RoME), emphasizing the importance of systematically evaluating modeling costs and potential returns to guide enterprise modeling initiatives effectively. In [27], the same authors proposed further research to refine RoME, highlighting

J. Grabis et al. (Eds.): RCIS 2025, LNBIP 547, pp. 103–118, 2025.
https://doi.org/10.1007/978-3-031-92474-3_7

the need for theory-driven and practice-oriented approaches to address trade-offs between modeling efforts and their potential returns within specific contexts and purposes. The investigation reported in [33] revealed that modelers rely on subjective assessments of the returns from their modeling activities, largely due to challenges in quantifying the benefits and costs associated with the modeling process or its outcomes. Building on these findings, recent work has explored pain points in domain modeling and their relation with modeling costs [32]. In this paper, we contribute to the RoME research agenda by proposing risk assessment to support decision-making in domain modeling. More precisely, we advocate a critical examination of models and the modeling process through a risk-oriented lens, drawing upon the Common Ontology of ValuE and Risk (COVER) [28] and the Reference Ontology for Security Engineering (ROSE) [23].

Risk assessment identifies and evaluates potential risks, providing the foundation for cost-benefit analysis to consider expected and unforeseen outcomes. By framing modeling practices as an ecosystem of objects, events, agents, and their capabilities and vulnerabilities, we can evaluate real and hypothetical scenarios to identify the risks involved in modeling practices. This approach allows identifying assets at risk to determine (a) when models mitigate risks (models as risk deterrents), (b) when they might introduce risks to other organizational assets (models as risk sources), and (c) when they are vulnerable to risk events (models as assets at risk) [23,28,29]. When models act as risk deterrents, they can help reduce costs; however, when they act as risk sources or assets at risk, they may incur additional costs.

Our proposal facilitates effective trade-off evaluations uncovering options with high benefits and manageable risks. For modelers and organizations, it can aid in assessing the benefits and costs of models and modeling practices, ensuring that modeling investments are aligned with practical constraints and organizational goals. It can also help researchers and developers identify gaps that can lead to improvements in modeling languages, methods, and tools.

To illustrate the practical application of our proposal, consider the healthcare sector, which faces significant data interoperability challenges due to the rapid development of ICTs. This results in heterogeneous data across systems, ranging from patient records to billing information. Differences in data formats and terminologies create interoperability barriers, driving up costs, increasing error rates, and potentially compromising patient safety. To address these risks, the European Reference Network for Rare Neuromuscular Diseases (EURO-NMD) developed the FAIR-by-design EURO-NMD Registry to built-in interoperability founded on using accepted ontologies and classifications that promote data integration [1]. This registry integrates clinician-entered data, patient-reported outcomes, and quality-of-life measures, exemplifying how conceptual models can serve as security mechanisms to mitigate risks and control costs.

To explore the feasibility of our proposal, we analyze cases that evidence the practical use of domain models. These cases are drawn from the literature, as well as nine semi-structured interviews with professionals and researchers active

in this field.[1] To the best of our knowledge, no existing literature addresses risks in conceptual modeling in the way proposed by this study. Instead, prior research largely focuses on the role of models as tools for managing risks in security [23], business [20], information technology [30], and other fields.

The rest of the paper unfolds as follows. Section 2 introduces our notions of domain models and risk. Section 3 presents an analysis of models as countermeasures, assets at risk, and threat objects. Section 4 makes some final considerations and discusses the direction of future work.

2 Background Knowledge

2.1 Structural Conceptual Models

The risk-oriented perspective proposed in this paper applies to various domain models. Nevertheless, this study focuses on its applicability in structural conceptual models, a category of domain models designed to represent a domain by capturing its structural regularities [15]. Such structural regularities involve types (classes), attributes, relationships, and constraints. As a result, they are versatile tools that serve a wide range of purposes, including problem-solving, decision-making, learning, and system development [33]. Examples of such models include those specified using Entity Relationship (ER) [7], UML Class Diagrams [24], OntoUML [15], and Object-Role Modeling (ORM) [16]. Henceforth, all references to "model(s)" or "modeling" in the remainder of this paper should be interpreted within the context of structural conceptual modeling.

2.2 Assumptions on the Nature of Risk

Despite considerable progress in clarifying the concept of risk, the term remains overloaded and conceptually ambiguous. Practitioners and researchers across various fields continue to define, perceive, and assess risks in different ways [2]. This study adopts the ontological assumptions about the nature of risk as defined by the Common Ontology of ValuE and Risk (COVER) [28]. The most relevant assumptions for this investigation are outlined below.

First, *risk is assessed in terms of its impact on goals*. For instance, if one is concerned with the risk of missing a train, this is because missing a train impacts one's goals, such as arriving on time at a meeting or saving money.

Second, *risk is experiential*, implying that risk is attributed to events rather than objects. In this view, a risk assessment targeting an object is driven by the overall risk from events that could affect it. For instance, consider the risks associated with a laptop. Assessing these risks involves examining: (i) the goals it enables (e.g., completing tasks, accessing documents), (ii) potential events that could disrupt these goals (e.g., hardware failure, data loss), and (iii) the circumstances that might lead to such events (e.g., accidental spills, power surges).

[1] These online interviews were conducted and anonymized by the primary author as part of a previous study detailed in [33].

Then the risk your phone is exposed to is the aggregation of the risk of hardware breakdowns, phishing attacks, and so on.

Third, *risk is contextual*, meaning that the risk an object is exposed to may vary even if its intrinsic properties remain constant. For example, while the characteristics of a car, such as an anti-lock braking system, can affect the risk of a car accident, external factors such as road conditions and weather conditions can increase or enable risks.

Finally, *risk is relative*, suggesting that an event may be perceived as risky by one agent but deemed non-risky by another. For instance, consider a hacker attack. From the target's perspective, the attack constitutes a risk to the security of their assets. Conversely, for the hacker, it represents a strategy to achieve their objectives rather than a risk.

2.3 Risk Terminology

Our analysis is based on a fragment of COVER [28] that addresses risk experience, which focuses on (unwanted) events and their underlying causes. We also incorporate the risk management perspective formalized by the Reference Ontology for Security Engineering (ROSE) [23]. From these ontologies, we adopt the following concepts:

- Intention: It is an intrinsic property characterized by a commitment to achieving a goal, where the goal represents the propositional content associated with this property. *Example:* Sarah's decision to attend the gym illustrates her intention to lose weight. *Note:* This definition closely aligns with the notion of an objective as defined in ISO 31073 [18].
- Agent: It is an entity with an Intention to achieve a specific goal within a given context. *Example:* A person committed to losing weight, or a company committed to increasing its profits.
- Threat Event: It is an event with the potential to cause loss or damage to an Agent's Intention. *Example:* An armed robbery threatens personal safety or property, while a factory fire threatens infrastructure and operations. *Note:* This definition closely aligns with the definition on NIST SP 800-30 Rev.1 [22] to the same term.
- Loss Event: It is an event that results in actual loss or damage to an Agent's Intention. *Example:* The theft of valuables during a robbery or the destruction of factory machinery in a fire.
- Risk Event: It is any event that can be categorized as a Threat Event or a Loss Event.
- Risk Experience: It is a complex event composed of Risk Events. *Note:* This definition closely aligns with the notion of a threat scenario as defined in NIST SP 800-30 Rev.1 [22].
- Risk Subject: It is an Agent whose Intention is at stake during an Loss Event. *Example:* A robbery victim at risk of losing safety or property, or a factory owner whose business is threatened by a fire.

- Object at Risk: It is an asset (person or thing) that may be harmed or damaged by a Loss Event. *Example:* The personal belongings of a robbery victim or the machinery within a factory that could be destroyed by fire.
- Threat Object: It is an object (person or thing) bearing capabilities that, when manifested, can cause a Threat Event. *Example:* The perpetrator responsible for committing a robbery or a short circuit in factory wiring that triggers a fire. *Note 1:* This concept is referred to as risk object in [5], hazard or risk source in ISO 31073 [18], and threat actor in NIST SP 800-221 [22].
- Risk Enabler: It is an ancillary object (person or thing) with dispositions that enable or increase the likelihood of a Risk Event. *Example:* The weapon used by a robber to escalate the threat or the presence of flammable materials in a factory that increases the likelihood of a fire.
- Threatening Situation: It is a circumstance or condition that favors the occurrence of a Threat Event. *Example:* Walking alone on a dark street late at night heightens the risk of robbery, just as neglecting wiring maintenance increases the likelihood of a fire.
- Security Mechanism: It is an object (person or thing) intentionally designed to create value by systematically protecting an Agent's Intention from Risk Events. *Example:* An anti-theft alarm protects an individual's intention to ensure safety by deterring theft attempts.
- Control Event: It is an event that directly or indirectly mitigates the occurrence of Risk Events. *Example:* A thief abandoning a robbery attempt upon hearing an alarm sound.
- Control Chain Event: It is an event that can cause Control Event. *Example:* An alarm emitting a loud sound as a warning of a theft attempt.
- Vulnerabilities: They are dispositions inhering in a Risk Enabler or an Object at Risk whose manifestations can triggers a Risk Event. *Example:* A house lacking an anti-theft alarm is more vulnerable and therefore more likely to be targeted by thieves. *Note:* This definition closely aligns with the definition on NIST SP 800-160v1r1 [22] and ISO 31073 [18] to the same term.

To represent these concepts clearly and facilitate the visualization of our analysis, we developed the diagrams presented in Sect. 3. Although we based these diagrams on the formalizations of COVER [29] and ROSE [23] and used symbols inspired by CORAS [21], they do not adhere to standard conceptual modeling conventions. This approach was driven by the lack of a formalized language capable of fully representing the COVER and ROSE concepts required for our analysis. As a result, we created a custom set of symbols that more effectively suit our goals.

3 Domain Modeling Through a Risk-Oriented Lens

In applying the assumptions and concepts discussed in Sect. 2 to domain modeling, we assert that models and elements of the modeling process are exposed to risk events that can affect the modeling goals proposed in [14] and validated

in [33]. The impact of these events is contingent on the context and perspective from which they are analyzed. Therefore, in this section, we analyze several practical examples.

3.1 Risks Mitigated by Models

Examining the scenarios experienced by the interviewees and the application of models in practice, as evidenced in [3,8–11,17,34], models can be conceptualized as Security Mechanisms in certain circumstances. As such, models can prevent Risk Events. For instance, an interviewee highlighted that an OntoUML model created by his development team to facilitate the implementation of three systems was subsequently used for training by the company's tax team. This use led to a more efficient training process, substantially reducing training costs. As shown in Fig. 1, in this scenario, the OntoUML model could be understood as a Security Mechanism to mitigate **the risk of lengthy and ineffective training of new employees**, which leads to higher training costs. In micro or small businesses, mitigating this risk alone may not justify the investment in developing a domain model. However, this interviewee works for a company that, by 2022, had a workforce of approximately 45,000 employees. Although the exact number of individuals who reaped the model's advantages remains undisclosed, given the size of this organization, one can imagine the financial impact that a reduction in training costs could have.

The prevalent use of models to support system development among the interviewees highlights the significant role of domain models as Security Mechanisms in software engineering. The results of the study conducted in [11] revealed that practitioners agreed regarding the effectiveness of conceptual models in facilitating various aspects of system development, including creating user-friendly systems, alignment with requirements, and adaptability to changing needs. In addition, they considered these models more effective than code in understanding system behavior and communicating system functionality to stakeholders. According to Liddle [19], model-driven methodologies potentially facilitate the early identification of defects (e.g. design flaws, omissions) and misunderstandings between stakeholders and developers. While adjustments can be made later in the development process to address oversights from the initial conceptualization phase, subsequent modifications become increasingly costly and disruptive as development progresses. Late-stage alterations incur greater expenses and yield less effective solutions. Consequently, in software engineering, model-centric approaches can leverage the model as a Security Mechanisms to mitigate **risks associated with technical challenges** such as domain debts [31], false agreement issues [13], and software entropy [6], **as well as economic concerns** like escalated rework costs, prolonged development timelines, and unnecessary expenditure of resources.

To illustrate the use of models as Security Mechanisms in software engineering, we will explore two examples retrieved from the interviews. In the first case, the interviewee was responsible for solving a system interoperability challenge. Specifically, the goal was to merge data from street cameras and a search

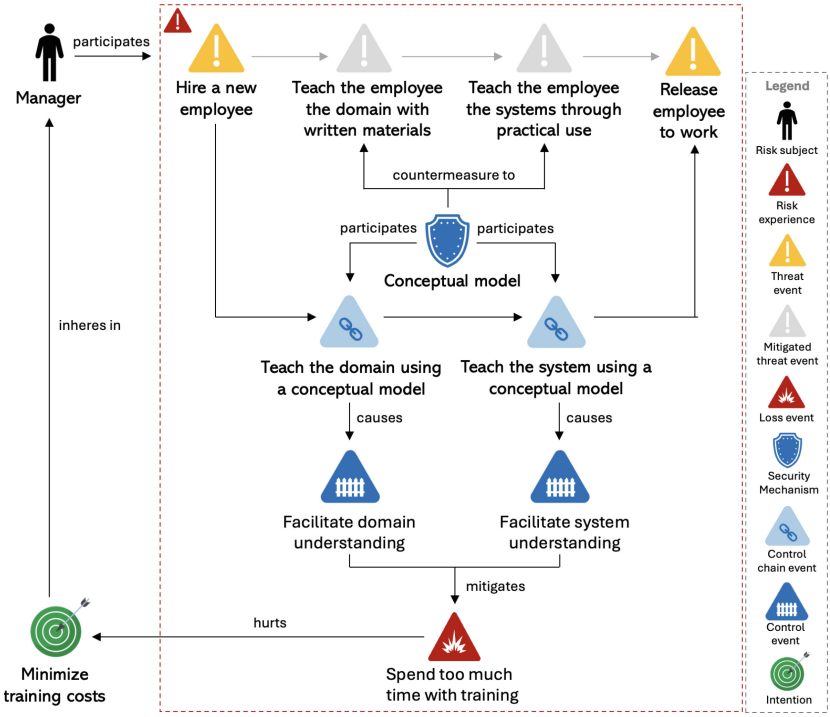

Fig. 1. Model as a Security Mechanism

engine to enable a police officer to run queries such as "How many people wearing black coats were in the downtown area of a given city on Christmas Eve morning?". However, his team encountered false agreement issues. An example of a false agreement in this context might occur when the police officer differentiates between coats and jackets when formulating search engine queries, while the camera system categorizes these terms identically in its recordings. To solve the problem, the interviewee developed conceptual models for both systems, which were crucial in mitigating **the risk of incorrect data integration**, thereby reducing rework, time, and costs in the process of systems interoperation.

In the second case, the interviewee was hired to document a system that was commercialized and maintained by a software company. The company used a UML model to facilitate system maintenance and to train novice programmers. On the one hand, the model potentially mitigated **the risks associated with direct changes to the code during system maintenance**. Such changes could lead to the risk of a trial-and-error cycle for developers, resulting in significant rework costs and a lengthy maintenance process. By using the model, developers could systematically analyze available options, perform testing, and select the optimal update path before implementing changes in the code. On the other hand, the model aided in mitigating risks that arose when a new program-

mer attempted to understand the system solely through practical use, such as **the risk of misunderstandings regarding data entry or relationships**. This method, while conventional, tends to be more time-consuming and error-prone than the alternative approach of studying the model and understanding how the system works analytically. Therefore, the model-driven approach was considered a safer and more cost-effective method in both situations.

3.2 Risks Created by Models

Upon reflection on the contexts in which models are employed, it becomes clear that certain risk events happen because of models. As the cases discussed in this session demonstrate, a model becomes a threat when it participates in risk events that could endanger other organizational assets.

Outdated models may be the source of several practical issues. For instance, the interviewee as cited in the preceding case, who developed a model for documenting an existing system, disclosed that the model became outdated upon his departure from the project. As depicted in Fig. 2, if programmers rely on this outdated model to learn about the domain covered by the system, they would be exposed to **the risk of an increased time to develop a new feature or fix a bug**, because of their inadequate understanding of the domain. As highlighted in [11], the fact that models become outdated and inconsistent with the underlying code represents a significant challenge in model-driven software development. For instance, using outdated models for maintenance exposes developers to **the risks of erroneous system updates, inadequate data integration, and incorrect domain interpretations**.

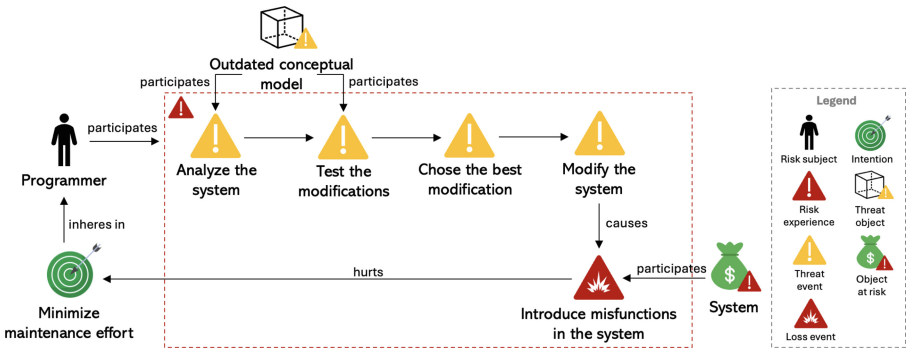

Fig. 2. Outdated Model as a Threat Object

Implementation-dependent models can be understood as a threat because of the foundational design choices embedded in them. As noted in one interview, and supported by the study conducted in [26], models can pose risks when their development prioritizes implementation details over domain understanding. A participant in Petre's study [26] mentioned that a drawback of using

UML models emerges when their use diverts attention away from the primary focus on notational concerns. This participant said, "You start making decisions about how you are going to implement classes, inheritance, etc., when you should be mapping out what their business does". Adopting an implementation-dependent model may expose programmers to **the risk of limited system maintenance and reuse** because this model lacks critical information needed for system updates, data integration, and systems interoperability. They can also result in accrued domain debt [31] during the development process or in systems that are either unusable by end-users or only partially satisfy stakeholder needs.

Inadequately constructed or maintained models can present risks that extend beyond the realm of systems development and maintenance; they can also affect scenarios involving, for example, data integration and knowledge acquisition. Consequently, when examining a model that lacks comprehensiveness, a data analyst may be exposed to the **risk of erroneously integrating data** because of misinterpretations of data relationships. Similarly, a manager may be exposed to **the risk of acquiring inaccurate domain knowledge** by relying on a poorly structured model to understand a domain.

While poorly constructed and maintained models are typically the primary source of risks, *well-developed and maintained models* can also contribute to risks under certain circumstances. For instance, they may expose organizations to the **risk of disclosing sensitive information**. Consider once again the situation in which the interviewee was hired to document an existing system that was commercialized and maintained by a software company. He revealed a frustration stemming from the company's reluctance to use the model as a communication medium with external stakeholders. However, he acknowledged the company's concern that the potential value of the model in this regard was outweighed by the associated risk of potentially revealing sensitive information, such as how the system implemented the domain constraints. This scenario is illustrated in Fig. 3, which demonstrates how the model can assume the role of a Threat Object in contexts where the information it contains is an asset to the organization.

Finally, *the modeling process* itself can be considered a threat in certain circumstances. For instance, organizations can be exposed to **the risk of an extensive investment of time and resources in creating or maintaining a model** that could jeopardize a project's feasibility. This argument is often used by those who oppose the practical use of models, particularly in software development [11,19].

3.3 Models at Risk

Our analysis of the contexts in which models are used reveals situations in which the model itself is at risk. This stems from the fundamental premise that the model's value depends on its accuracy, completeness, usability, availability, and up-to-dateness while emerging from its use. Consequently, any circumstance that poses a risk to those qualities can be understood as a threat. When such risks

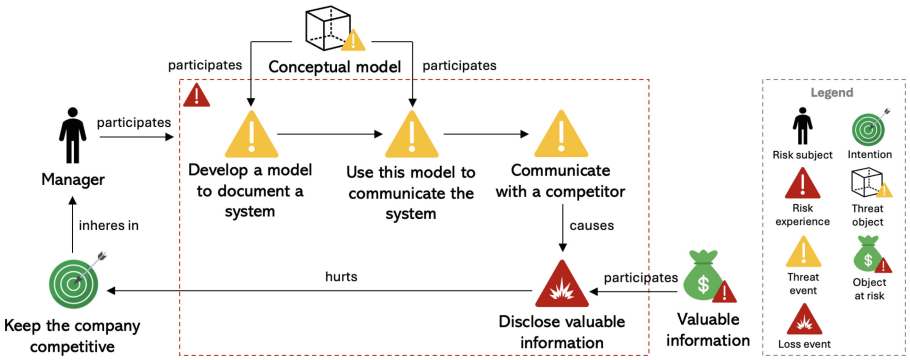

Fig. 3. Valuable Model as a Threat Object

materialize, the model will fail to meet the goals set by the user or the modeler. In the rest of this section, we discuss some risks models may be exposed to.

Neglecting model maintenance poses a significant threat that can lead to **the risk of losing control over model updates**. According to Valle [33], such occurrences are prevalent in projects undertaken by organizations lacking a modeling culture. Given that modeling expertise typically resides with the modeler, the model tends to be neglected once their involvement concludes. As a result, the model may become outdated, reducing its effectiveness and usefulness in various activities. As discussed in Sect. 3.2, outdated models have the potential to precipitate substantial practical issues, contributing to erroneous decision-making or flawed knowledge acquisition. Therefore, an outdated model has minimal utility.

The unavailability of domain experts is another potential threat that can affect model use and reuse. Figure 4 illustrates a scenario in which this situation was a threat to the project managed by one interviewee. He worked in the healthcare domain and had limited access to domain experts to validate the model. Because of this limitation, his team had to validate the model based on their own knowledge and information obtained from technical reports and the company's website. This led to uncertainties regarding the model's quality and usability, thereby increasing **the risk of the model being unsuitable for use or not being used at all**. The interviewee said: "I had to admit that if we would want to really know if the model would apply to the health care institutes, we should do some extra research." Because of this issue, his team started a new project centered on improving model communication and comprehension, in which they developed a thesaurus to define the domain's key concepts. While necessary and beneficial to facilitate later validation of the model with domain experts, this additional effort required extra investments.

Inexperienced modelers are a potential threat to models and modeling effectiveness, as their lack of modeling skills can increase **the risk of models becoming incorrect or difficult to understand and use**. For instance, two interviewees highlighted the importance of a good diagram layout to increase

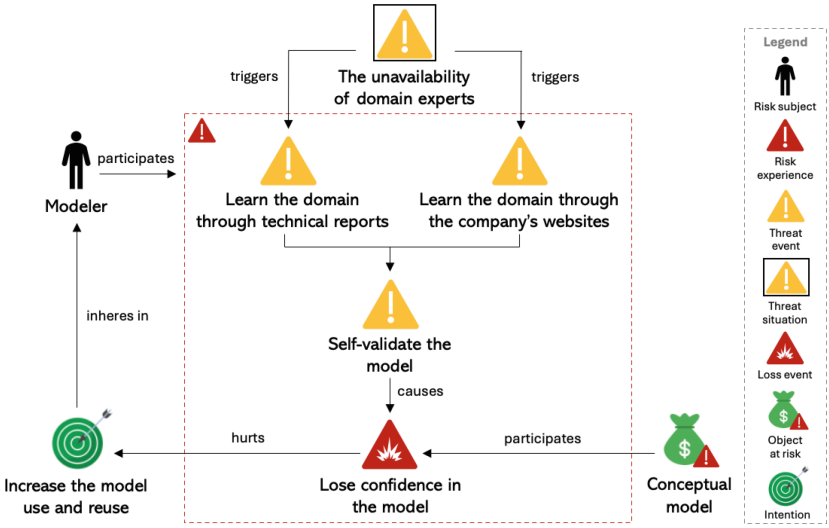

Fig. 4. Model as an Object at Risk with a Threatening Situation example

the model's comprehensibility. One of them said: "For people that are not used to reading models, I am sure that the position, the presentation, and the colors are really important. Even so important that they will say that they like it very much or don't like it at all." Therefore, the modeler's lack of familiarity with diagram layout standards [4] can lead to the creation of visually inappropriate diagrams, resulting in models that are difficult to interpret and understand. For example, diagrams with crossed lines, curved lines, or overlapping classes.

Users with limited modeling experience can also pose a threat, as they may lack the necessary skills to understand and use the model appropriately. As a result, there is **the risk of models being used incorrectly or for purposes inappropriate to their design**. In such circumstances, users may perceive the model as inadequate to meet their needs, thereby diminishing its value. For instance, individuals attempting to use an ER model for process understanding may experience frustration and fail to achieve their goals, possibly leading to the conclusion that the model is ineffective. However, while an ER model excels at conceptualizing data-oriented domains, its applicability may be limited in more process-oriented domains, where a process model would be a more appropriate alternative. To mitigate the risks emerging from users lacking modeling skills, one interviewee said that he devised training courses to instruct users on understanding and utilizing the OntoUML model developed by his team. Moreover, they progressively engaged these users in the process of model creation.

Model complexity poses a threat when models reach a level of complexity where the effort required for their development or maintenance outweighs their potential benefits. As outlined in [12], the maintainability of UML class diagrams correlates directly with the complexity of the models. This complexity

can be measured by factors including the total number of classes, associations, aggregations, attributes, and similar elements, as well as the organization and comprehensibility of these elements. One interviewee, who had developed a model to guide the development of a system, mentioned during the interview **the risk of the model becoming obsolete** if it became too complex because of the high cost and effort involved in using and maintaining it. He said: "The model generates a cost over time, right? In other words, when the model starts to get too complex, people do not want to work on the model. They think it will be more work to tinker with the model than to do it without a model."

Limited modeling tools also may pose a threat to models. For instance, if a modeler manually verifies the model, there is a possibility that certain errors will escape detection. Consequently, the resulting model derived from such verification would be exposed to **the risk of being semantically unreliable**, rendering any subsequent use prone to errors. As shown in Fig. 5, if manual verification becomes necessary due to the limitations of the modeling tool to support automated verification, it can be argued that the modeling tool contributed to the occurrence of verification errors by assuming the role of Threat Enabler. One interviewee mentioned another instance where modeling tools could also be considered a Threat Enabler facilitating the exposure of models to **the risk of having a poor layout**. According to him, "Tools are very bad at getting the model layout right. Getting the lines the way you want them. Doing all that seems like such a silly thing, something that the tools should have fixed by now, but they haven't." Beyond the tool itself, inappropriate modeling methods or languages can also act as potential Threat Enablers facilitating the occurrence of certain Risk Events.

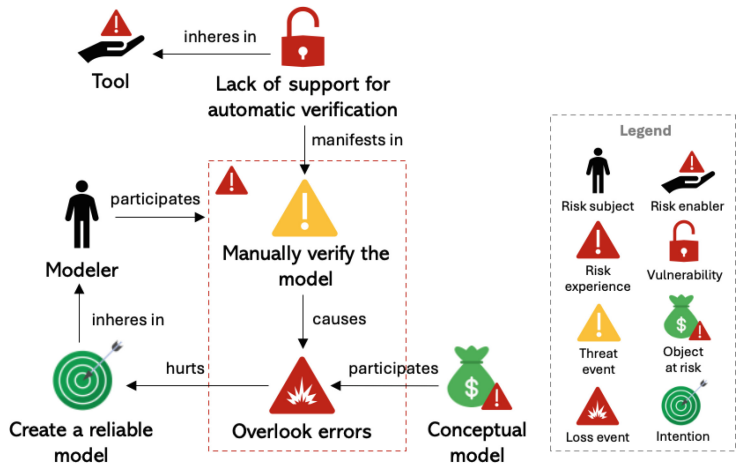

Fig. 5. Model as an Object at Risk with a Risk Enabler example

The risks inherent in insufficient model assessment are a concern highlighted in [25]. Model and its constraints require validation[2] and verification[3] before implementation, as this practice can prevent numerous design flaws and implementation errors. Particularly for mission-critical systems, error detection during modeling can prevent failures that could lead to significant damage. While various approaches to model validation and verification attempt to mitigate the associated risks, these tasks remain inherently risky and dependent on external factors such as the model's quality, the modeler's expertise, and the modeling tools' capabilities.

4 Final Considerations

Managing models from a risk-centric perspective is essential for analyzing model benefits and costs, planning modeling investments, and integrating modeling practices across projects and organizations. Treating models as assets facilitates evaluating the cost-benefit ratio and return on modeling efforts for model-based solutions, guiding the allocation of resources to increase model value while reducing risk and cost. Accordingly, investments should focus on enabling models to serve as countermeasures against risk events and mitigating risks that could undermine the value of models or those introduced by the models themselves.

Building on the foundations of COVER [28] and ROSE [23], we identified that models can assume three distinct roles: as a Security Mechanism, where its capabilities, when manifested, prevent risk events from occurring (e.g., using data models to avoid system interoperability issues); as a Threat Object, where its capabilities, when manifested, lead to risk events that could jeopardize goals (e.g., using an outdated model to guide system maintenance, resulting in incorrect decision-making); and as an Object at risk, where the model's vulnerabilities can be manifested in threatening situations composing the modeling process (e.g., using a modeling tool that lacks layout functionalities, which may result in poorly designed diagrams and reduce the model's comprehensibility). Figure 6 presents a non-exhaustive summary of examples from this paper, illustrating circumstances under which models assume these roles.

Thinking of a model as a Security Mechanism helps one to identify the various situations in which it is useful to mitigate risk and avoid additional costs. It encourages modelers and organizations to think about the model beyond its original development objectives. This broader perspective enhances the model's value and helps amortize its development costs. For instance, if developing a model M to achieve objective O1 costs C, and the same model M can later be used to achieve objectives O2 and O3 without further investment, the model's overall value increases while its development cost is diluted. Therefore, this strategy is also a valuable tool for convincing skeptical sponsors and coworkers of the merits of investing in modeling efforts.

[2] Have we built the correct model, according to the pre-specified requirements?.
[3] Have we built the model correctly, without syntactic and semantic flaws?.

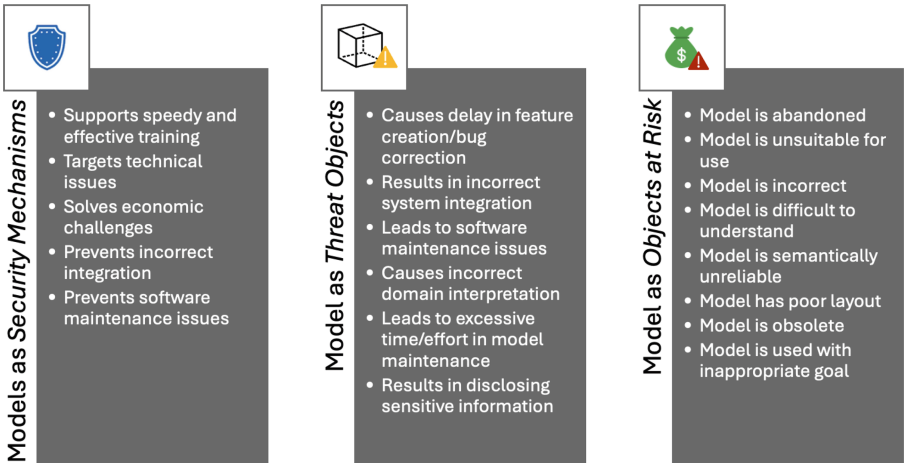

Fig. 6. Non-exhaustive summary of examples illustrating circumstances under which models play the role of Security Mechanisms, Threat Objects, or Objects at Risk

Posing models as Threat Objects allows one to identify circumstances in which they should be avoided because of their potential to create more problems than solutions. Models become threats when they participate in risk events that could endanger other organizational assets, such as systems or sensitive information. Although threats often stem from poorly constructed and maintained models, well-developed models can also pose risks under certain circumstances. Therefore, it is imperative to ensure that the appropriate model is used for the intended purpose.

Viewing a model as an Object at risk facilitates the identification of threats that may undermine its value and increase its costs. These costs may arise either from complications inherent in the modeling process or from challenges encountered in using the model. The risk perspective provides valuable insights into the vulnerabilities of the model, the modeling process, the modeler, and the tools and languages used. By addressing these vulnerabilities, modelers and organizations can improve model quality and better plan modeling initiatives, by taking proactive measures to mitigate the risks of failure or adverse outcomes. Additionally, their identification could assist researchers in refining modeling languages, methods, tools, and training programs.

Through this study, we learned about several gaps that could be exploited to enhance the efficacy of modeling initiatives. Our risk-centered analysis underscored opportunities for improving modeling tools, languages, and methods that developers and researchers can explore. Additionally, it revealed alternative approaches and perspectives that organizations and modelers can consider when planning and executing their modeling efforts. We also recognize that risk analysis centered on models and the modeling process is not trivial, underscoring the potential value of a dedicated language to guide such analysis.

Future research could investigate how the risks identified in this study can be translated into quantifiable costs, and how the costs of domain models change over the life cycle of the model. In addition, an analysis of the risks and costs associated with not using these models in practical projects could be conducted. Finally, the formalization of a structured language tailored to modeling risk and security situations would be beneficial.

References

1. Atalaia, A., et al.: EURO-NMD registry: federated FAIR infrastructure, innovative technologies and concepts of a patient-centred registry for rare neuromuscular disorders. Orphanet J. Rare Dis. **19**(1), 66 (2024)
2. Aven, T., et al.: On the ontological status of the concept of risk. Saf. Sci. **49**(8–9), 1074–1079 (2011)
3. Batra, D., Marakas, G.M.: Conceptual data modelling in theory and practice. Eur. J. Inf. Syst. **4**(3), 185–193 (1995)
4. Bergström, G., et al.: Evaluating the layout quality of UML class diagrams using machine learning. J. Syst. Softw. **192**, 111413 (2022)
5. Boholm, Å., Corvellec, H.: A relational theory of risk. J. Risk Res. **14**(2), 175–190 (2011)
6. Canfora, G., et al.: How changes affect software entropy: an empirical study. Empir. Softw. Eng. **19**, 1–38 (2014)
7. Chen, P.: The entity-relationship model: toward a unified view of data. ACM Trans. Database Syst. **1**(1), 9–36 (1976)
8. Davies, I., et al.: How do practitioners use conceptual modeling in practice? Data Knowl. Eng. **58**(3), 358–380 (2006)
9. Dobing, B., Parsons, J.: How UML is used. Commun. ACM **49**(5), 109–113 (2006)
10. Fettke, P.: How conceptual modeling is used. Commun. Assoc. Inf. Syst. **25**(1), 43 (2009)
11. Forward, A., Lethbridge, T.C.: Problems and opportunities for model-centric versus code-centric software development: a survey of software professionals. In: MISE, pp. 27–32 (2008)
12. Genero, M., et al.: Assessing object-oriented conceptual models maintainability. In: Advanced Conceptual Modeling Techniques: ER 2002 Workshops, pp. 288–299. Springer (2003)
13. Guarino, N.: Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), vol. 46. IOS press (1998)
14. Guizzardi, G., Proper, H.A.: On understanding the value of domain modeling. In: Proceedings of 15th VMBO (2021)
15. Guizzardi, G., et al.: Towards ontological foundations for conceptual modeling: the unified foundational ontology (UFO) story. Appl. Ontol. **10**(3–4), 259–271 (2015)
16. Halpin, T.A., Morgan, T.: Information Modeling and Relational Databases. Morgan Kaufmann, Burlington (2010)
17. Ho-Quang, T., et al.: Practices and perceptions of UML use in open source projects. In: International Conference on Software Engineering, pp. 203–212 (2017)
18. ISO: ISO 31073:2022 - Risk management – Vocabulary (2022)
19. Liddle, S.W.: Model-driven software development. In: Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges, pp. 17–54. Springer (2011)

20. Lu, Y., Marais, K.B., Zhang, S.G.: Conceptual modeling of training and organizational risk dynamics. Procedia Eng. **80**, 313–328 (2014)
21. Lund, M.S., et al.: A guided tour of the CORAS method. Model-driven risk analysis: The CORAS approach, pp. 23–43 (2011)
22. National Institute of Standards and Technology (NIST): NIST Computer Security Resource Center (CSRC) Glossary (2025)
23. Oliveira, Í., et al.: An ontology of security from a risk treatment perspective. In: Conceptual Modeling. ER 2022, vol. 13607, pp. 365–379. Springer, Cham (2022)
24. OMG: UML 2.0 superstructure specification – Final adopted specification. Technical report ptc/03-08-02, Object Management Group (August 2003)
25. Pakalnickiene, E., Nemuraite, L.: Checking of conceptual models with integrity constraints. Inf. Technol. Control **36**(3) (2007)
26. Petre, M.: UML in practice. In: 2013 35th International Conference on Software Engineering (ICSE), pp. 722–731. IEEE (2013)
27. Proper, H.A., Guizzardi, G.: Modeling for enterprises; let's go to RoME ViA RiME. In: The Practice of Enterprise Modeling 2022 Forum, vol. 3327 (2022)
28. Sales, T.P., et al.: The common ontology of value and risk. In: Conceptual Modeling: 37th International Conference, ER, pp. 121–135. Springer (2018)
29. Sales, T.P., et al.: Ontological analysis and redesign of risk modeling in archimate. In: 22nd EDOC, pp. 154–163. IEEE (2018)
30. Strecker, S., Heise, D., Frank, U.: RiskM: a multi-perspective modeling method for IT risk assessment. Inf. Syst. Front. **13**, 595–611 (2011)
31. Störrle, H., Ciolkowski, M.: Stepping away from the lamppost: domain-level technical debt. In: 45th SEAA, pp. 325–332 (2019)
32. Valle, I., et al.: Unraveling the pain points of domain modeling, Preprint on SSNR at https://ssrn.com/abstract=5045287
33. Valle, I., et al.: What do I get from modeling? An empirical study on using structural conceptual models. In: Enterprise Design, Operations, and Computing, pp. 21–38. Springer, Cham (2023)
34. Verbruggen, C., Snoeck, M.: Practitioners' experiences with model-driven engineering: a meta-review. Softw. Syst. Model. **22**(1), 111–129 (2023)