Computer Science Review 50 (2023) 100597

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review article

Model-based joint analysis of safety and security: Survey and identification of gaps☆

Stefano M. Nicoletti [a,*], Marijn Peppelman [a], Christina Kolb [b], Mariëlle Stoelinga [a,c]

[a] University of Twente, Formal Methods and Tools, Enschede, The Netherlands
[b] University of Twente, Industrial Engineering & Business Information Systems, Enschede, The Netherlands
[c] Radboud University, Department of Software Science, Nijmegen, The Netherlands

ARTICLE INFO

Keywords:
Safety
Security
Model-based
Fault trees
Attack trees
Component fault trees
Attack-fault trees
State/event fault trees
BDMPs
Bow ties
SysML
STAMP
ALLOY
Event-B
Bayesian networks
AADL

ABSTRACT

We survey the state-of-the-art on model-based formalisms for safety and security joint analysis, where safety refers to the absence of unintended failures, and security to absence of malicious attacks. We conduct a thorough literature review and – as a result – we consider fourteen model-based formalisms and compare them with respect to several criteria: (1) *Modeling capabilities and Expressiveness:* which phenomena can be expressed in these formalisms? To which extent can they capture safety-security interactions? (2) *Analytical capabilities:* which analysis types are supported? (3) *Practical applicability:* to what extent have the formalisms been used to analyze small or larger case studies? Furthermore, (1) we present more precise definitions for safety-security dependencies in tree-like formalisms; (2) we showcase the potential of each formalism by modeling the same toy example from the literature and (3) we present our findings and reflect on possible ways to narrow highlighted gaps. In summary, our key findings are the following: (1) the majority of approaches combine tree-like formal models; (2) the exact nature of safety-security interaction is still ill-understood and (3) diverse formalisms can capture different interactions; (4) analyzed formalisms merge modeling constructs from existing safety- and security-specific formalisms, without introducing *ad hoc* constructs to model safety-security interactions, or (5) metrics to analyze trade offs. Moreover, (6) large case studies representing safety-security interactions are still missing.

1. Introduction

New technology comes with new risks: self-driving cars or train automation systems [1] may get hacked, people depend on the proper functioning of medical implants for their continued health [2]. Such risks concern both accidental failures (safety) and malicious attacks (security). Safety and security can be heavily intertwined. Measures that increase safety may decrease security and vice versa: the Internet-of-Things offers ample opportunities to monitor the safety of a power plant, but their many access points are notorious for enabling hackers to enter the system [3]. Passwords secure patients' medical data, but are a hindrance during emergencies. This is not an entirely new problem [4]: in fact, it has been widely acknowledged – including by international risk standards [5,6] – that safety and security must be analyzed in combination [7,8]. To cater for this need, various risk frameworks have been developed for the joint analysis of safety and security, i.e. the safety-security co-analysis. *Process-oriented* frameworks consider the steps needed for performing safety-security risk analysis [9]. Various formalisms specifically support the *risk assessment phase* within this process, i.e., the identification, analysis and evaluation of safety-security risks. Text-based methods include FMVEA [10,11], CHASSIS [10,12], HAZOP [13,14] and SAHARA [15].

Motivation. Our focus, however, is on *model-based* risk assessment. Model-based formalisms provide a detailed insight in different failure and attack modes, mechanisms and their root causes. Understanding these is crucial for effective decision making. Furthermore, model-based formalisms allow one to compute various dependability metrics: such metrics quantify key performance indicators, such as the system availability and mean time to attack/failure. The goal of this paper

☆ This work was partially funded by the NWO, The Netherlands grant NWA.1160.18.238 (PrimaVera), and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101008233, and the European Research Council Consolidator Grant 864075 (*CAESAR*).

* Corresponding author.
E-mail addresses: s.m.nicoletti@utwente.nl (S.M. Nicoletti), m.peppelman@utwente.nl (M. Peppelman), c.kolb@utwente.nl (C. Kolb), m.i.a.stoelinga@utwente.nl (M. Stoelinga).

https://doi.org/10.1016/j.cosrev.2023.100597
Received 7 November 2022; Accepted 17 October 2023
Available online 7 November 2023

is to survey the state-of-the-art on these model-based formalisms for safety-security co-analysis, as an overarching work unitedly addressing this literature space is still lacking. We compare these formalisms with respect to several criteria: (1) *Modeling capabilities and Expressiveness:* which phenomena can be expressed in these formalisms? To which extent can they capture safety-security interactions? (2) *Analytical capabilities:* which analysis types are supported? (3) *Practical applicability:* to what extend have the formalisms been used to analyze small or larger case studies? These questions summarize well known criteria used when selecting appropriate formal methods for a given task, and synthesize principles seen in corresponding literature [16].

Methodology: an overview. To address the proposed task, we conducted a systematic and standard literature review process, which yielded 14 formalisms for safety-security co-analysis (see Table 1). We grouped these into 4 categories: *Combinations of attack trees and fault trees*; *Formalisms extending attack trees and/or fault trees*; *Mathematical formalisms* and *Architectural formalisms* that extend architectural system models with additional means for safety-security aspects. For *expressiveness*, we compare to what extent these formalisms are able to capture the four safety-security interactions identified by Kriaa et al. [17]: *Conditional dependency*, where security requirements necessitate safety requirements, or vice-versa; *Mutual reinforcement*, where safety requirements or measures increase security, or vice-versa; *Antagonism*, where safety and security requirements or measures conflict with each other; and *Independence*, where no interaction takes place. To illustrate what the formalisms look like, we model in each formalism the Locked Door Example [18,19]. This is a classical example of safety-security interaction, exemplifying an antagonistic dependency between safety and security: if locked, a door is unsafe in case of a fire. However, if unlocked, the door is insecure in case of a burglary. Our methodology is thoroughly presented in Section 2.

Key findings. Our survey revealed several noteworthy findings, some of which are summarized in Tables 2 and 3. First, most formalisms are based on extensions of Attack Trees (ATs) and Fault Trees (FTs). Since FTs and ATs are widespread formalisms that model how systems can fail or be attacked, this is not surprising. Extensions of these formalisms are Attack-Fault Trees (AFTs) [20], Component Fault Trees (CFTs) [21,22], Fault Tree/Attack Trees (FT/ATs) [23], Boolean driven Markov processes (BDMPs) [24], Attack Tree Bow Ties (ATBTs) [25,26], Failure-Attack-CounTermeasure (FACT) Graphs [27], and State/Event Fault Trees (SEFTs) [28]. Secondly, the investigated formalisms can capture different safety-security interactions. These formalisms are mostly based on safety-specific or security-specific techniques, that are further adapted to account for the other. However, none of the formalisms introduce novel modeling constructs to capture safety-security interactions, and neither are novel metrics introduced to capture dependencies between safety and security. Moreover, some methods can only provide qualitative insight into the safety-security relationships, while others can also account for quantitative (failure/attack) probabilities. Furthermore, a large-sized industrial case study has not yet been conducted. Finally, some of the investigated modeling methods could be enhanced to increase their expressiveness. We detail these findings and reflect on highlighted gaps in Section 10.

Related work. There are a few earlier surveys [38] on safety-security analysis methods. An important survey was published by Kriaa et al. [17] that introduces definitions for safety and security interactions. The authors compare non-model-based and model-based approaches, the latter being graphical or non-graphical. The survey [39] compares seven frameworks with respect to model creation, origin, stages of the risk assessment, and main use cases. The comparison conducted here is, however, carried out at a high level. The survey [40] summarizes current practices in safety and in security modeling. Then a new model is proposed, combining the goal structuring notation GSL with Attack-Defense Trees. In contrast to their work, we focus on model-based approaches and we present more detailed definitions for safety and security interactions. Furthermore, we employ a running example that we model in every analyzed formalism whenever possible. We then compare formalisms with respect to their ability to model safety-security interactions. Our work details how the formalisms work, what their technical capabilities are, and how safety-security interact. The survey [41] provides an overview about text-based models for engineering. It investigates research questions such as at which development stage the research was conducted, which methods and tools were employed during the research — e.g., FMVEA [10,11], CHASSIS [10,12], HAZOP [13,14] and SAHARA [15] — what the classification of the contribution of the research is, in which research domains were the results evaluated, where was the research published, and what the research publication time-line and trend is. In contrast to their work, we focus our attention on model-based formalisms rather than on text-based ones, and we are interested in concrete dependencies between safety and security. Furthermore, we present examples for each formalism. Finally, the survey [42] provides an overview of different kinds of formalisms that consider safety or security, although separately. On the contrary, our focus is on formalisms that consider *both* safety and security.

Table 1
Overview of safety-security formalisms.
Source: Citations from Google Scholar, October 2023.

Formalism	Ref.	Year	#Citations
Fault Tree/Attack Tree Integration (FT/AT)	[23]	2009	210
Component Fault Trees (CFTs)	[22]	2013	79
Attack-Fault Trees (AFTs)	[29]	2017	109
State/Event Fault Trees (SEFTs)	[28]	2013	34
Failure-Attack-CounTermeasure (FACT) Graphs	[27]	2015	104
Boolean Driven Markov Processes (BDMPs)	[19]	2014	53
Attack Tree Bow-ties (ATBTs)	[25]	2018	177
Bayesian Networks (BNs)	[30]	2015	64
Threats-Hazards-Opportunities (THO) Framework	[31]	2007	388
STAMP	[32]	2017	253
SysML	[33]	2011	104
ALLOY	[34]	2002	1736
Event-B	[35]	2017	20
Architectural Analysis and Design Language (AADL)	[36]	2020	5

A summary of our contributions:

(1) We conduct a thorough literature review on model-based formalisms for joint analysis of safety and security.

(2) We compare found formalisms by their modeling and analysis capabilities, their application domains in industry and expressiveness, i.e., their ability to capture safety-security dependencies.

(3) We present more precise definitions for safety-security dependencies in tree-like formalisms (Section 6).

(4) We showcase the potential of each formalism by modeling the same toy example from the literature [18,19].

(5) We present our findings and reflect on possible ways to narrow highlighted gaps (Sections 10 and 11).

Structure of the paper. Section 2 details our methodology, Section 3 provides background for fault trees and attack trees., Sections 4, 5, 7 and 8 compare the various formalisms. Section 6 presents our definitions for dependencies on tree-like formalisms, Section 9 presents a comparison of formalisms based on their expressiveness. Section 10 presents – and reflects on – our findings and Section 11 concludes the paper.

2. Methodology

Research questions. The goal of this survey is to understand the state-of-the-art on safety-security co-analysis and to identify future needs. In

Table 2
Comparison of safety-security formalisms. A = Antagonism, CD = Conditional Dependency, MR = Mutual reinforcement, I = Independence. T = Time/order, S = States, C = Consequences. ∗ = capable when NOT-gate is supported. → = capable but only directional from security to safety. ∘ = only if BNs are dynamic. □ = possible in principle, not showcased in literature.

Formalism	Dependencies				Expressiveness			Modeling	Application	Tool
	A	CD	MR	I	T	S	C			
FT/AT	$\overset{*}{\rightarrow}$	→		x				ATs refine FT leaves	Chemical plant	
CFTs	∗	x	x	x				Merge ATs + FTs	Cruise control	SafeTbox [37]
AFTs	x	x	x	x	x			Merge dynamic ATs + FTs	Pipeline, lock door	UPPAAL
SEFTs	x	→	x	x		x		FTs + Petri nets	Tyre pressure, lock door	ESSaRel
FACT graphs	$\overset{*}{\rightarrow}$	→		x	x			FT/ATs + Triggers + Countermeasures	Overpressure vessel	
BDMPs	x	x	x	x	x	x		Triggers, Petri nets	Pipeline, lock door	KB3, Figaro
ATBTs	$\overset{*}{\rightarrow}$	→		x			x	Bowties + FT/AT	Pipeline, Stuxnet	
BNs	x	x	x	x	∘	∘	x	Conditional prob.	Pipeline	MSBNx
THO framework	$\overset{*}{\rightarrow}$	→		x			x	FTs + event trees	Electrical grid	
STAMP			x					Process controller	Synchronous-islanding	
SysML								System components	Embedded systems	TTool
ALLOY	x	x	x	x		x		Prove system correctness	Fire detection system	ALLOY
Event-B	x	x	x	x	x	x	□	Prove system correctness	Charging system	RODIN
AADL			x					System components + ports	Lock door	Cheddar, Marzhin

Table 3
Overview of analysis capabilities of safety-security formalisms. QL = Qualitative Analysis, QT = Quantitative Analysis.

Formalism	Analysis		Details
	QL	QT	
FT/AT	x	x	Same as base FTs (MCS, MPS, probabilities...).
CFTs	x	x	Same as base FTs (MCS, MPS, probabilities...) + extension on MCS analysis.
AFTs	x	x	Same as base FTs + time, cost, likelihood of an attack. Trade offs between attributes.
SEFTs	x	x	Same as FT/ATs plus details on the state of components via Petri nets.
FACT Graphs	x	x	Same as FT/ATs plus triggers and countermeasures
BDMPs	x	x	Mean Time to Success, probability of success, list of possible attack success sequences.
ATBTs	x	x	Risk level evaluation like regular BTs, trade offs analysis.
BNs	x	x	Calculate reliability metrics (mean time to failure) and conditional independence analysis.
THO Framework	x	x	Likelihood of an attack
STAMP	x		Identify potential hazards and undesired behaviors.
SysML	x		Checks for reachable states that violate safety properties/security requirements.
ALLOY	x		Prove properties of a system under given assumptions.
Event-B	x		Prove properties of a system under given assumptions.
AADL	x	x	Calculate MTBFs of the system. Error model statements can generate FTs.

particular, when analyzing surveyed formalisms, we focus on the following questions: (**Q1**) How expressive are these formalisms, e.g., how many safety-security interaction can they capture? (**Q2**) Which modeling constructs exist to model the interactions between safety and security? (**Q3**) Which analyses do these formalisms enable? (**Q4**) How do these formalisms compare on industrial case studies and (**Q5**) What are the gaps and the findings? What would be desirable extensions? These questions summarize well known criteria employed to assess formal methods, and synthesize principles seen in corresponding literature [16].

Search methodology. We followed an established methodology for our literature search [43,44]. Our aim was to find peer-reviewed publications from 1922 until 2020, focusing on model-based formalisms jointly analyzing safety and security. Following [43], we first queried the publication databases Scopus, Microsoft Academic and Google Scholar with relevant keywords. By using "safety" and "security" we collected 4997 results. We further refined them – 2714 entries – by focusing on "model-based" formalisms and subsequently by employing keywords such as "dependency" together with appropriate Boolean operators and wildcards. Finally, we refined our search by exploring the formalisms and tools we found (e.g., fault-attack trees, BDMP, AADL). We also explored earlier surveys on safety-security combinations [17,39,40]. Moreover, we considered several safety-only and security-only analysis frameworks, to see if they had been extended to handle safety-security combinations. In particular, we looked into threat analysis [45,46]: however, to the best of our knowledge, no threat analysis framework tackles safety-security interactions in a model-based fashion. After gathering selected literature, publications were independently assessed by three reviewers following the methodology showcased in [44]. They were included or cited only upon agreement of two or more reviewers. Papers that were included in our final selection are listed in Table 1, together with number of citations taken from Google Scholar. The following paragraph showcases literature that was considered but then excluded from our final selection.

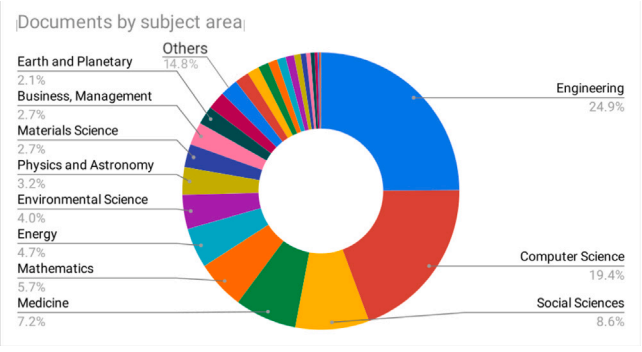

Fig. 1. Percentages of published work that mention "safety" and "security" in title, abstract or keywords, divided by research field (time frame: 1922–2020).
Data source: Scopus.

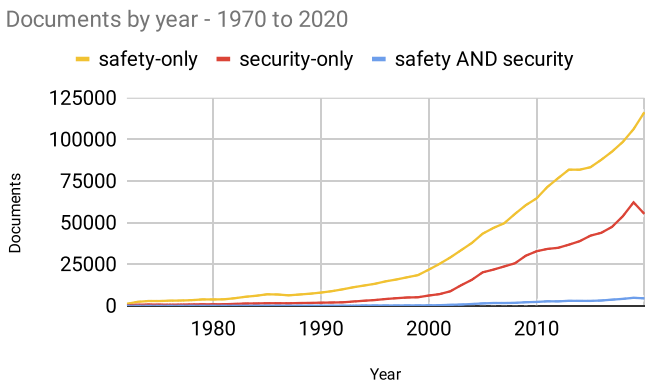

Fig. 2. Number of published documents with "safety" (yellow line in the figure), "security" (red line) and both "safety" and "security" (blue line) in title, abstract or keywords, ranging from 1970 to 2020.
Data source: Scopus.

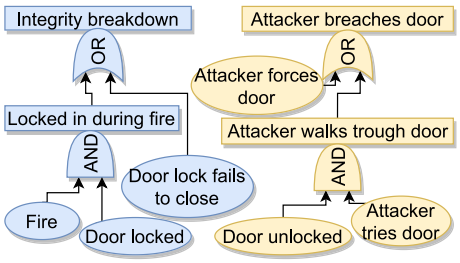

Fig. 3. Fault tree (on the left) and attack tree (on the right). Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) events.

Excluded papers. During our literature review, we encountered papers on related topics regarding safety-security, which do not match the focus of this survey. We encountered a few papers on S-Cube [47,48]. This modeling technique is very relevant for modeling safety and security interactions, however it seems to be focused on SCADA systems specifically. A paper on SOTERIA [49] presents modeling of IoT device networks for automated safety and security analysis from the device source code. This modeling relates specific to IoT devices. In this survey, we are interested in more general modeling techniques. There were several methods based on textual approaches, such as CHASSIS [10,12], FMVEA/FMEA [10,11,50] and SAHARA [15]. However, they are based on a textual structure, while this survey focuses on model based approaches.

Bibliometric analysis. The areas of computer science and engineering cover about 44.4% of the published research on safety and security in the considered time frame, thus being the two major fields interested in this topic (as seen in Fig. 1).

Moreover, by comparing the bibliometric data regarding literature about safety, security and both safety and security we uncovered a growing interest in these fields since 1970s, testified by an increasingly higher number of publications mentioning these terms in the abstract, title or keywords. As shown in Fig. 2 the amount of publications that mention both safety and security in these fields is also increasing, while remaining at a considerably lower number when compared with safety-only and security-only literature. While the interest for their interactions is rising, the low amount of publications on safety and security suggest that there is still a considerable amount of work to be done in order to address challenges and open problems in the area of safety-security co-analysis (cfr. the blue line in Fig. 2).

3. Background on attack trees and fault trees

Since several safety-security formalism combine attack trees and fault trees, we briefly introduce these formalisms before discussing their combinations. In Section 4, we compare plain combinations of Fault Trees (FTs) and Attack Trees (ATs), while Section 5 surveys FTs and ATs combinations with additional features.

Modelling. FTs and ATs are hierarchical diagrams that model how low level failures (resp. attacks) propagate through the system and lead to system level failures (resp. attacks). Despite their name, FTs and ATs are Directed Acyclic Graphs (DAGs), rather than trees, since subtrees can have multiple parent gates. Furthermore, FTs are part of international standards [51]. FTs were developed in the early '60s [52] alongside Fault Tree Analysis (FTA) [53] and, due to their popularity, ATs were proposed in the '90 as their security counterparts [54]. FTs (resp. ATs) start with a Top Level Event (TLE), modeling a system level failure (attack), which is then refined through Boolean gates: the AND-gate indicates that all children must fail (be attacked) in order for the gate to fail (be attacked). For the OR-gate to fail (be attacked), at least one of its children need to fail (be attacked). When refining is no longer needed, one arrives at leaves of the tree: the Basic Events (BEs) in FTs model atomic failures; the Basic Attack Steps (BASs) in ATs model atomic attack steps. Non-leaves nodes (Fig. 3), such as *Locked in during fire* and *Attacker walks through door*, are called *intermediate events*, denoted by rectangles. The FT in Fig. 3 models that integrity is compromised if one is either locked in during a fire or the door lock fails to close. This is modeled via the top OR-gate. One is locked in during a fire if there is a fire and the door is locked. This is modeled via the AND-gate connecting the BEs *Fire* and *Locked*.

Analysis. FT and AT enable different kinds of analyses [55]: *qualitative analyses* include Minimal Cut Sets (MCSs), indicating which combinations of BEs or BASs lead to the TLE. The set {Fire, Door locked} is a cut set in Fig. 3. Quantitative analyses compute *dependability metrics*, such as the system reliability, attack probabilities and costs. For example, by equipping the BEs and BASs with probabilities, one can compute the likelihood of a system level failure or attack to occur. FTs and ATs have been used to analyze numerous case studies [56–58].

Observations. Apart from similarities, FTs and ATs also feature some remarkable differences: FTs often focus on probabilities, whereas ATs consider several other attributes, like cost, effort and required skills [59]. The difference with respect to the OR-gate between FTs and ATs is that in FTs, the probability of failure asks for total probability. Thus the (probability) value of an OR-gate is the sum of the values of its children, minus the value of their intersection. In ATs instead, attacks are characterized by their max probability. So the value of an OR-gate in ATs is the max value among its children. Furthermore, FTs have been extended with repairs [60], and dynamic gates [61,62]; ATs with defenses, and sequential AND (SAND) gates [57,63].

4. Category 1: formalisms combining fault trees and attack trees

We first survey formalisms that combine FTs and ATs: *Fault Tree/Attack Trees* refine the basic events of a fault tree by an attack tree; *Component Fault Trees* merge fault trees and attack trees; and *Attack-Fault Trees* merge dynamic fault trees and dynamic attack trees. In all figures, we use blue for safety-events, yellow for security, and brown for their combinations. In the following paragraphs, we highlight answers to research questions with bold placeholders, e.g., (**A1**) denotes an answer to the first research question for the considered formalism.

4.1. Fault tree-attack trees

Modelling. Fault Tree/Attack Trees (FT/ATs) [23] are based on the assumption that attackers try to force a system failure by triggering the BEs in a FT. Thus, (**A2**) a FT/ATs indicates how the BEs can be

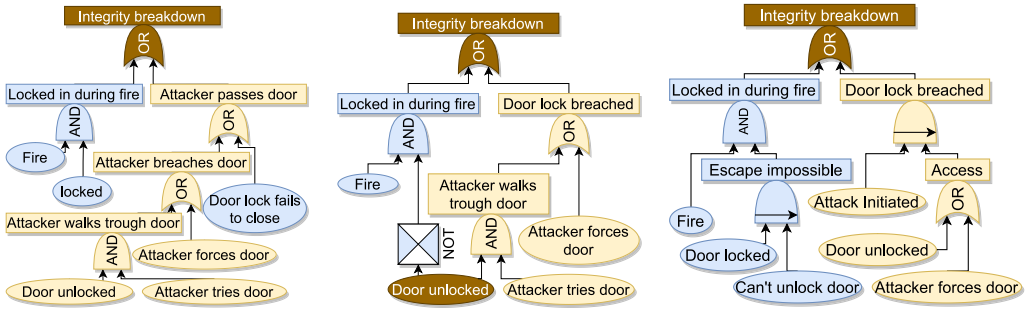

Fig. 4. Lock door example as (a) FT/AT (left), (b) component fault tree (center) and (c) attack-fault tree (right). Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) events. Nodes in brown indicate events related to both safety and security.

attacked by refining these BEs by ATs with the BE as goal. By making this BE the goal of the AT depicted in yellow, Fig. 4(a) details the basic event *Door lock fails* in Fig. 3. (**A1**) FT/ATs do not provide a clear method for modeling dependencies of components in the AT on parts of the FT, so mutual reinforcement cannot be modeled due to a lack of bidirectionality. Conditional dependency and antagonism can be modeled as per Section 6.

Analysis. The paper [23] (**A3**) computes the probability for the TLE to occur, given probabilities for the BEs and the BASs.

Observations. Exploiting safety faults is a common method for hackers to enter a system, e.g.,: triggering a fire alarm to disengage a fire safety lock. Thus, it seems reasonable to consider the leaves of a FT as vulnerabilities, refining them via ATs. The fault tree's TLE then becomes the target for hackers. This can be reasonable e.g., shutting down a factory for ransomware. However, attackers are often exploiting safety faults to achieve other goals, such as stealing digital assets (e.g., Stuxnet worm targeting the availability of nuclear power plants to damage the nuclear program of Iran [64]). In that case, the forcing of BEs is only a starting point in an attack. (**A5**) Subsequent attack steps could be modeled in an AT; this could be a natural extension for FT/ATs. (**A4**) FT/ATs are used to model a case study on toxic chemical spill at a chemical plant in [23].

4.2. Component fault trees

Modelling. Component Fault Trees (CFTs) equip FTs with a modular structure [21], so that a large FT can be modeled and analyzed in terms of smaller components. (**A2**) The paper [22] extends CFTs with security aspects by introducing a new BEs type for security breaches, which are essentially BASs. CFTs make no distinction between BEs and BASs. This is exemplified in Fig. 4(b), where attacks and failures are freely merged.

(**A1**) As per Section 6, CFTs model mutual reinforcement, conditional and antagonistic dependencies. In particular, antagonism is achieved by connecting one event A to an intermediate safety event B and an intermediate security event C, but having one of those connections be through a not gate (see Fig. 5).

Analysis. (**A3**) The CFTs from [22] enable the same analysis methods as FTs. The authors remark that, if a BAS only occurs in MCSs with multiple, low probability and independent BEs, then this attack is very unlikely to ever cause a disruption, as it requires multiple other unlikely events to occur. However, one may remark that the same holds for any event.

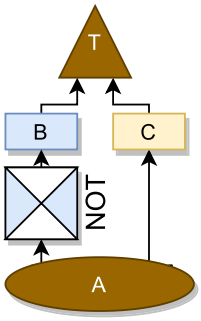

Fig. 5. CFT antagonism. Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) events. Nodes in brown indicate events related to both safety and security.

Observations. (**A5**) Not distinguishing between safety and security has the advantage that all existing tools remain applicable. Moreover, one may ask if it really matters whether a disruption is due to a failure or an attack. A disadvantage of merging failures and attacks, however, is that no interactions or trade offs between safety and security can be studied. (**A4**) Paper [22] models an Adaptive Cruise Control system as a case study.

4.3. Attack-Fault Trees

Modelling. Attack-Fault Trees (AFTs) [29] treat attacks and failures in the same way CFTs do. However, (**A2**) AFTs merge *dynamic* ATs and *dynamic* FTs. These provide additional gates to model dynamic behavior: dynamic ATs include the sequential AND-gate (SAND), modeling attacks as sequence of steps [20]. Dynamic FTs include gates for modeling spare components, functional dependencies (FDEP) and priority ANDs. Further, attacker profiles quantify the BASs with attacker's capabilities, such as resources, skills and damage. Fig. 4(c) presents the Locked Door Example as an AFT. The various SAND-gates indicate the order of events. E.g., for the *Door lock breached* event to happen, first *Attack initiated* must happen, and then the *Access* event as well.

(**A1**) AFTs can model conditional dependencies more explicitly via the FDEP gate: the trigger of the FDEP (i.e. leftmost input) automatically makes the dependent events (i.e. the other inputs) fail or be attacked. Even though AFTs do not include a NOT-gate, antagonism between events A and $\neg A$ can be expressed through IFAIL nodes A and $\neg A$, where A has probability 0 if the $\neg A$ is activated, and vice versa (Fig. 6).

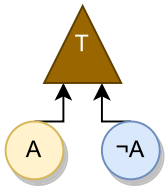

Fig. 6. AFT antagonism. Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) events. Nodes in brown indicate events related to both safety and security.

Analysis. (**A3**) AFTs are analyzed via statistical model checking by translating the AFTs to a network of stochastic timed automata. Using the attribute values in the attacker profiles, several metrics can be computed, such as the time, cost and likelihood of the attacks. Pareto frontiers elucidate trade offs between attributes, e.g., the likelihood of an attack within a given budget.

Observations. (**A5**) AFTs share the two disadvantages common to all FT approaches in that there is no distinction between safety or security failures once the TLE occurs, and the methods of modeling interactions may not clearly highlight the antagonistic dependencies. The attacker profiles in AFTs support a wide range of quantifiable parameters, enabling versatile analysis and trade offs. (**A4**) Remarkably, AFTs are used to model the medium-sized case study of an oil pipeline, in addition to modeling the Locked Door Example.

5. Category 2: formalisms extending fault trees and/or attack trees

This section surveys combinations of FTs and ATs with additional features: *State/Event Fault Trees* join a fault tree-like model with Petri nets, *FACT Graphs* join FTs and ATs with countermeasures and the ability to capture dynamic behaviors using triggers, *Boolean Driven Markov Processes* extend FTs and ATs with Petri nets and triggers, and *Attack Tree Bow-ties* combine Event Trees with an FT/AT-like model.

5.1. State/Event fault trees

Modelling. (**A2**) State/Event Fault Trees (SEFTs) [28] join FTs and Petri nets,[1] expressing that certain failures can only happen in certain states. Whereas the leaves in ATs and FTs model atomic events, SEFTs deploy Petri nets to accommodate state changes inside basic events. In Fig. 7, the *Door* component can move between the states *Unlocked* and *Locked*. These state changes are triggered by events, depicted as black rectangles, that can be exponentially distributed, deterministic, or triggered by other events. Both states and events can be communicated via the gates of the tree, via in and out ports. In this way, Fig. 7 expresses that a fire casualty can only happen if the door is locked. Fig. 7 models antagonism: the door should be unlocked to escape from *Fire* but locked to prevent failure of the AND-gate *Enter unlocked door*. Following [28], we would model each subsequent attack step in the attacker component: first, the attacker tries to enter the door and, if it is not open, he/she tries to force it (the *Force door* step). However, to better represent antagonism wrt. the *Door* component, we decided to model the *Attacker tries door* step by embedding an AND-gate in the *House* component.

(**A1**) Besides antagonism, SEFTs support (directional) conditional dependency and mutual reinforcement, as per Section 6. This is achieved by a state machine that will always activate a safety or security state (Fig. 8). (**A5**) Reworking SEFTs so that conditional dependency from safety to security can also be accounted could address a gap in the formalism and be considered a desirable extension.

[1] [28] says state charts instead of Petri nets, but we did not see any state chart constructs, like hierarchical composition.

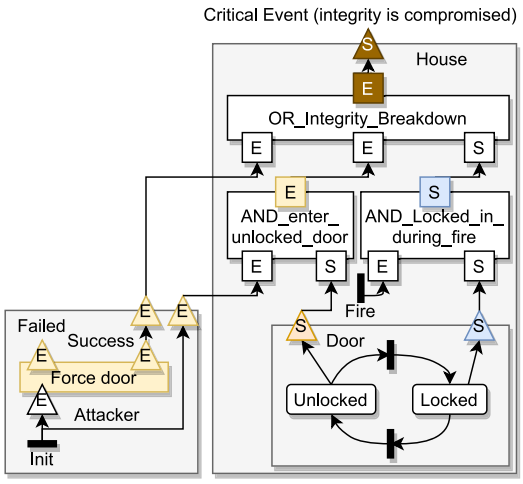

Fig. 7. State-event fault trees. Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) states/events. Nodes in brown indicate states/events related to both safety and security.

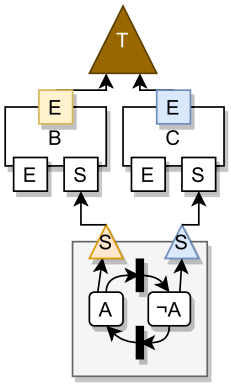

Fig. 8. SEFT antagonism. Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) states/events. Nodes in brown indicate states/events related to both safety and security.

Analysis. (**A3**) SEFTs support the same dependability metrics as attack-fault tree combinations. The tool ESSaReL translates SEFTs to extended deterministic stochastic Petri nets, which can be further analyzed by the TimeNet tool, e.g., for steady state analysis.

Observations. (**A4**) The authors of [28] model the small-sized example of a Tyre Pressure Monitoring System. Even though not mentioned explicitly in [28], it appears that Petri nets in the SEFT leaves must be disjoint between components. E.g., it is not possible to use the *Unlocked* state of the *Door* component as a direct input for an attack step. Allowing this could increase the expressivity of SEFTs.

5.2. Failure-Attack-CounTermeasure (FACT) graphs

Modelling. Failure-Attack-CounTermeasure (FACT) Graphs [27] (**A2**) join FTs and ATs with countermeasures and the ability to capture dynamic behaviors using triggers. To capture failures, attacks, and possible countermeasures the authors provide as a first step importing FTs at the end of the safety hazard and risk assessment phase. In the second step safety countermeasures are added to the graph. The third step adds an AT to the graph. Attacks are related to failures: much like FT/ATs, the elements of the FT are detailed with an OR-gate to specify that the failure can happen by itself or through the action of an attacker. In step four security countermeasures are added, possibly to any element of the AT. This results in a model similar to an FT/AT, with added countermeasures, as illustrated in Fig. 9.

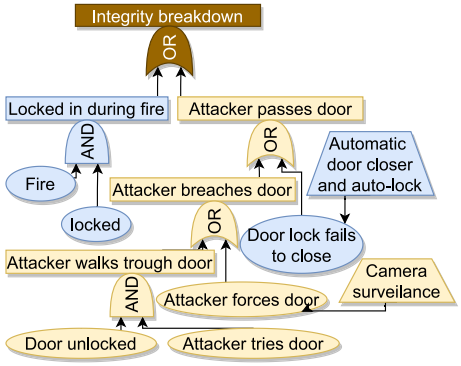

Fig. 9. FACT graphs. Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) events. Nodes in brown indicate events related to both safety and security.

Analysis. (**A3**) In spite of [27] not presenting any analysis of the proposed model, FACT graphs support analysis techniques like the ones enabled by FT/ATs. In addition to these, the presence of triggers and countermeasures could enable the analysis of the dynamic behavior of the system and the role of countermeasures.

Observations. (**A1**) Similarly to FT/AT, FACT graphs can capture conditional dependency only directionally: in fact, FTs are detailed with ATs, but not vice-versa. (**A5**) Antagonism could be easily obtained in case a NOT-gate is added, filling a sensible gap. In [27], (**A4**) the authors focus on cyber–physical systems (CPS) and model vessel overpressure with a FACT graph.

5.3. Boolean driven Markov processes

Modelling. (**A2**) Boolean Driven Markov Processes [24] extend FTs by equipping each leaf with a Markov process (MP), representing the different modes a component can be in. Various templates provide standard MPs to model standard failure behavior. For example, the *failure in operation* MP contains two modes, *operational* and *failed*. One transitions from *operational* to *failed* with an exponential failure rate λ, and back to operational with a repair rate μ. The IFAIL MP models instantaneous failures. Moreover, users can define their MPs as a stochastic Petri net [19], similarly to SEFTs. In [65], Boolean driven Markov processes (BDMPs) are extended with security aspects by providing additional Markov processes for attacker steps. For example, the *Attacker Action* MP (AA) contains tree modes: in *Idle* the attacker has not yet initiated the attack. The *Active* mode corresponds to actual attempt, requiring an exponentially distributed time to succeed, and leading to the *success* mode. Further, *triggers*, represented by dotted red arrows, allow one MP to trigger a mode change in another MP. Fig. 10 shows the Locked Door Example from [19]. The triggers pointing from *Attack initiated* to the OR-gate means that the OR-gate is not activated until *Attack initiated* happens. (**A1**) The use of Petri nets and the presence of intermediate events allow BDMPs to model all dependencies between safety and security, as detailed in Section 6. Antagonism is achieved by assigning the same Petri net to two leaves, activating one node on one state, and the other when not in that state (Fig. 8) (see Fig. 11).

Analysis. (**A3**) BDMPs allows both quantitative and qualitative analysis. The authors of [65] build BDMPs with the KB3 modeling software platform. The computation of the overall mean time to success (MTTS), the probability of success in a given time and the list of possible attack success sequences (ordered by decreasing probability) is possible.

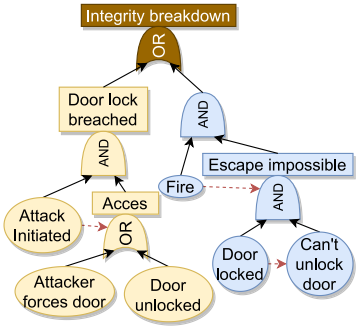

Fig. 10. BDMP. Dashed lines in red denote triggers. Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) events. Nodes in brown indicate events related to both safety and security.

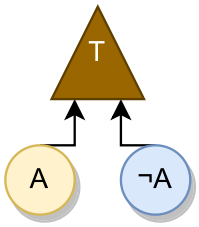

Fig. 11. BDMP antagonism. Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) events. Nodes in brown indicate events related to both safety and security.

Observations. [66] is (**A4**) one of the few analyzed papers with a medium-sized case study, referring to an oil/gas pipeline. Furthermore, the authors discuss dependencies corresponding to those mentioned in the introduction. (**A5**) As an observation, in BDMPs negation is not explicitly encoded in the structure but in underling Petri Nets for the leaves. An extension indicating such a negation similarly to how triggers are shown would be desirable. In Section 6 we propose a more general definition of dependencies in tree-like formalisms, thus dependencies definitions presented in [66] do not coincide with ours.

5.4. Attack Tree Bow-ties

Modelling. (**A2**) Attack Tree Bow Ties (ATBTs) [25,26] combine Bow-ties with ATs. Bow-ties [67] themselves combine FT and Event Trees (ETs): the left part of a Bow-tie is a FT modeling the causes of an hazardous event, which is in the middle of the Bow-tie. The ET on the right models its consequences. Barriers, i.e., a measure M preventing some failure F from happening, are modeled as AND(F, M), so that the failure F only propagates if the barrier M fails. Now, ATBTs extend regular Bow-ties by attaching ATs to the basic events of a FT, just as in FT/ATs. Thus, in ATBTs, the left part of the Bow-tie is an FT/AT, rather than an FT. Fig. 12 models the Locked Door Example via ATBTs. The leftmost part is equivalent to the FT/AT from Fig. 4(a). The ET on the right details the consequences in several cases: if the failure was a fire or a burglary, if alternate escape routes were available or inner doors were locked. (**A1**) Dependencies expressed in ATBTs are the same as in FT/ATs, as further explained in Section 6.

Analysis. (**A3**) Just like for regular Bow-ties, [25,26] identify vulnerabilities via MCSs. By assigning likelihood level to all BEs and BASs, two likelihood levels are assigned to these cut sets: one for safety and one for security. Thus, trade offs can be made.

Observations. Since the left, causal part of ATBTs is similar to FT/ATs, similar observations apply. (**A5**) Furthermore, it would be natural to study trade offs between safety and security by equipping ATBTs with

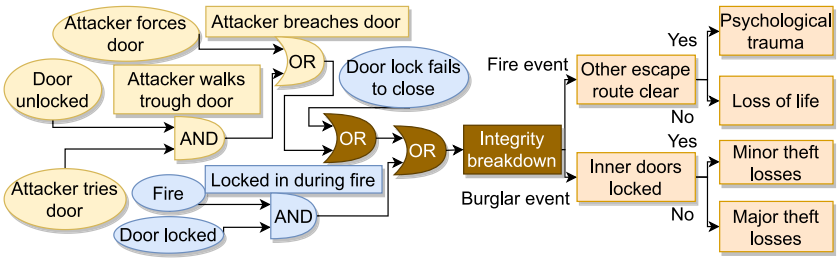

Fig. 12. Attack tree bowties. Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) events. Nodes in brown indicate events related to both safety and security. Nodes in orange indicate consequences.

two hazardous events, one for safety and one for security. Moreover, ATBTs are one of the few formalisms that consider safety-security trade offs. (**A4**) In [25,26] the authors create a small case study of a risk scenario in a chemical facility. In [68], the authors investigate the Facebook DNS outage from 2021 to show safety-security dependencies. They extend disruption trees which were introduced by [69] and add barriers (safety) and mitigations (security) in an attempt to model the four dependency types.

6. Dependencies in ATs and FTs combinations

Below, we propose more precise definitions of the dependency types from [17] for the specific context of FTs and ATs. We focus on *events* in FTs, ATs and their combinations. These events correspond to requirements: the TLE provides the main disruption to be avoided; the main requirement. The TLE refines into sub-events to be prevented; the sub-requirements. As such, these events and their interactions are the inverse of the logical interactions between the requirements. For each dependency type, we investigate to what extent these are expressible in the various FT and AT formalisms.

Antagonism. Two undesirable events A, B are *antagonistic*, or *conflicting*, with respect to C if at least one of them always occurs due to C. In tree-based formalisms, let A and B have a shared security/safety event C as a child. Let C be connected to either A or B through a NOT-gate. C (not) occurring will either trigger A or B directly, or the other through the NOT-gate.

The events *Locked in during fire* and *Attacker walks through door* in Fig. 4(b) are antagonistic with respect to *Door unlocked*. As such, expressing antagonism requires a form of negation. CFTs use NOT-gate to express negation. AFTs do so by tweaking (in a somewhat artificial way) the parameters of IFAIL. SEFTs and BDMPs express antagonism through a state-based model, where a system can be in only one state, e.g., door open or door closed. Like standard FTs and ATs, the FT/ATs and ATBTs models cannot model antagonism. However, a NOT-gate could easily be added.

Conditional dependency. If an undesirable event B is conditionally dependent on event A, then event B not occurring is only possible if event A has not occurred: A occurring implies B occurring. In tree-based formalisms, if we make B the TLE, and A a leaf, each set containing A must be a cut set for B. In Fig. 4(b), *Attacker forces door* is a condition for *Integrity breakdown*. All tree-based safety-security formalisms can express conditional dependencies between events. Since FT/ATs and ATBTs refine the leaves of a FT by an AT, they contain only paths from security to safety events. Thus, safety requirements can depend on security, but not the other way around. That also holds for SEFTs, where attack steps cannot depend on system's components that capture safety events. Further, the Functional Dependency (FDEP) gate in AFTs supports conditional dependencies, where B occurs as soon as A does.

Mutual reinforcement. Event A reinforces event B if the consequences of B are less likely to happen due to event A. In tree-based formalisms, event A reinforces event B, if every time B appears in a cut set, A does so as well. Events A and B mutually reinforce if the reverse also holds. Mutual enforcement typically occurs due to AND-gates, where both A and B are exclusively connected to the same AND-gate, either directly or through other (S)AND-gates. This configuration is expressible in CFTs, AFTs, SEFTs and BDMPs.

Independence. Two events A and B are statistically independent if $P[A\&B] = P[A]\cdot P[B]$. By assumption, all leaves in FTs and ATs are statistically independent. Events that are (mutually) reinforcing can also satisfy this statistical independence requirement. Thus, in tree-based formalisms the absence of (mutual) reinforcement is also required to capture Independence as intended. All tree-based formalisms from Sections 3 to 5 can express independence.

7. Category 3: Mathematical formalisms

Two formalisms are based on modeling outcomes and interactions primarily mathematically, namely Bayesian Networks and an unnamed framework based on Calculating the impact of Threats, Hazards, and Opportunities, which we will call the THO framework. The THO Framework allows analysis of threats, hazards and opportunities of a given system and presents similarities with ATBTs.

7.1. Bayesian Networks

Modelling. (**A2**) A Bayesian Network (BN) is a probabilistic graphical model that represent probabilistic dependencies between several variables via a directed acyclic graph. Each node A represents a variable, and an edge from A to B indicates that A stochastically depends on B. A conditional probability table yields the conditional probabilities $P[A|B]$. If the probabilities of the leaves in the BN are known, the probabilities of the root nodes can be calculated. In [30], BNs are proposed to model safety and security dependencies. The two root nodes represent system safety and security. Fig. 13 expresses that *Door locked* is a common factor for safety and security. (**A1**) BNs model conditional dependency, mutual reinforcement, and antagonism with a stochastic

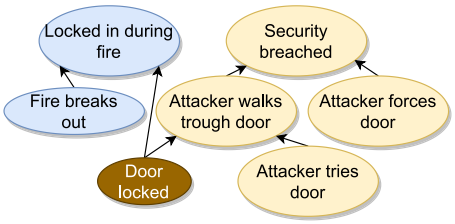

Fig. 13. Bayesian networks. Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) events. Nodes in brown indicate events related to both safety and security.

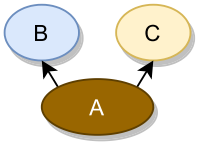

Fig. 14. Bayesian network antagonism. Nodes in blue (resp. yellow) indicate safety-related (resp. security-related) events. Nodes in brown indicate events related to both safety and security.

dependency in the DAG and appropriate values in the Conditional Probability Tables (CPTs). Antagonistic dependencies in particular are modeled by having a safety B and a security node C both depend on A, where the failure probability for one is higher when A is true, and for the other is higher when A is false: $P[B = 1|A = 1] > P[B = 1|A = 0] \wedge P[C = 1|A = 0] > P[C = 1|A = 1]$ (Fig. 14).

Analysis. (**A3**) Qualitative analysis can be done in the form of conditional independence analysis (analyzing which nodes influence other nodes and how). BNs enable quantitative analysis to calculate reliability metrics such as mean time to failure.

Observations. Fault trees and attack trees can be seen as special cases of Bayesian networks, where the CPT encode the Boolean gates, e.g., $P[A = 1|B = 1, C = 1] = 1$, 0 for other B and C, for an AND-gate A with children B and C. Thus, the BNs extend ATs and FTs with flexible dependencies, and enable separate TLEs for safety and security. Furthermore, events after a TLE failure could potentially also be modeled. [31] discusses modeling Bowties with conditional probabilities, which appears to be a perfect example of such an extension. However, the gates in ATs and FTs provide a clearer visualization of the behavior, since in BNs these must be read from the probability tables. (**A4**) Paper [30] uses the pipeline example as a case study. (**A5**) BNs can also be extended to Dynamic Bayesian Networks (DBNs), where multiple copies of the BN represent the state at different time steps. Nodes in the DBN can depend on nodes in previous time steps. More complex gates like sequence enforcers can then be modeled [70].

7.2. Threads-hazards-opportunities framework

Modelling. [31] presents a framework to perform risk analysis for both safety and security. When modeling a given system, the framework considers possible threats and hazards that can lead to various consequences, or *outcomes*. Opportunities – such as e.g., a planned shutdown, which allows for preventive maintenance – are also considered. Consequences would be typically expressed by real values representing *observable quantities* for e.g., economic loss, number of fatalities, number of attacks, the proportion of attacks being successful. The author provides steps that describe the risk and vulnerability analysis process: 1. Identify the relevant functions and subfunctions to be analyzed, and relevant performance measures (observable quantities). 2. Define the systems to meet these functions. 3. Identify relevant sources (threats, hazards, opportunities). 4. Perform an uncertainty analysis of the sources 5. Perform a consequence analysis, addressing uncertainties. 6. Describe risks and vulnerabilities. 7. Evaluate risks and vulnerabilities. 8. Identify possible measures, and return to 3. After identifying the relevant functions and subfunctions to be analyzed (Step 1), the system in question is defined (Step 2): understanding how the system works is a key step, so departures from normal, successful operation can be easily identified. In Step 3 threats, hazards and opportunities are identified. This can be done e.g., through analysis of statistics or through tools such as FMEA/FMECA/FMVEA [10,11,50] and HAZOP [14]. This step is heavily integrated with Step 4, performing an uncertainty analysis of sources. Once initiating events are identified and attackers' resources are assessed, event trees are used to develop scenarios starting from the initiating events. (**A2**) Once specific scenarios are identified e.g., a burglar entering the house, standard analysis can be performed using event trees and FTs (Step 5). In Step 6 risks and vulnerabilities are described. Specific quantities are selected e.g., the number of future attacks, the proportion of the attacks being successful, the number of successful attacks, and then uncertainties are assessed using probabilities: this leads to probability distributions of the above quantities (details on this in the following paragraph). Finally, Step 7 and 8 take place: however, [31] does not provide further details on these.

Analysis. (**A3**) For the quantitative analysis the uncertainties are expressed with probabilities and expected values $E(X)$ for the uncertainty distribution of X. For example X can take one of the values 0, 1, 50 and the associated probabilities are 0.8, 0.11 and 0.05, then the expected value is $E(X) = 0 \cdot 0.8 + 1 \cdot 0.11 + 50 \cdot 0.05$. In this framework, probability is used as a measure of uncertainty, seen through the eyes of the assessor. In contrast to BNs, when [31] uses the notation $P(C|D)$, it does not indicate the probabilities of C depend on the result of D. The probability assignments are dependent on available information and knowledge of the system. $P(C|D)$ indicates that the Expected value of C depends on D with unchanged probabilities. For example with sufficient information we are able to predict with certainty the value of the quantities of interest. The quantities are unknown to us as we have lack of knowledge, i.e., how people act, how machines work, etc. With the help of this framework one can calculate the likelihood of an attack and its consequences to failures.

Observations. (**A4**) In [31], the authors utilize the THO Framework to model the case study of an electrical grid. (**A1**) Given that [31] proposes to utilize both FTs and event trees, and to assess the impact of vulnerabilities/attacks on a given system, a clear parallel with ATBTs can be drawn. FTs can be employed to assess hazards on a given system, and we could evaluate the role of attacks by using ATs as seen in e.g., FT/ATs. Moreover, for every successful attack, consequences can be evaluated through the use of event trees. This structure suggests strong similarities with ATBTs. Thus, the framework proposed in [31] seems to capture dependencies in the same ways as ATBTs do (this also applies to our running example). (**A5**) It is however unclear if developing an ATBT for every identified scenario would be the best choice to represent the aforementioned procedure. Other options would include the development of an ATBT with multiple TLEs in order to represent diverse scenarios and their consequences.

8. Category 4: Architectural formalisms

Several of the formalisms are based on modeling the system architecture and then verifying its correctness, namely STAMP, SysML, Alloy, Event-B, and AADL. These models primarily describe the overall structure, architecture, behavior or interactions of the system they model, and then check if the modeled properties comply to a set of requirements and/or conditions.

8.1. STAMP

Modelling. The System-Theoretic Accident Model and Processes (STAMP) is rooted in the observation that system risks do not come

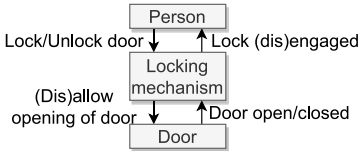

Fig. 15. STAMP.

from component failures, but from inadequate control or enforcement of safety-related constraints. Rather, (**A2**) in STAMP, systems are viewed as interrelated components that are kept in a state of dynamic equilibrium by feedback loops of information and control [71]. Each component enforces the safety and security constraints in the processes it controls, using control actions and feedback messages. Inability to enforce these constraints results in failures in safety or security. System Theoretic Process Analysis (STPA) and its extensions STPA-sec [72] and STPA-safesec [32] systematically identify the consequences of incorrect control actions and feedback [73], e.g., when these happen too early, in the wrong order, or were maliciously inserted. Fig. 15 shows the *Person*, who can lock and unlock the door. *Locking mechanism* is controlled by *Person*, and controls if the *Door* can open. A safety constraint is that *Person* must be able to unlock a door in case of fire; a security constraint is an unauthorized person must not be able to gain access. A *violation* is, e.g., the scenario where the person locks the locking mechanism while the door is open, forcing the door to stay open and granting unauthorized access. (**A5**) STPA-safesec can discover these risks in a structured manner, however this is currently still a manual process requiring domain knowledge and has not been automated [74]. (**A1**) STAMP is not geared towards expressing dependencies. However, STPA-safesec analysis may reveal safety-security conflicts [75].

Analysis. (**A3**) STPA is used to identify potential hazards and undesired behaviors: e.g., the violation previously described. It would for example invert the order of control actions, such as locking the lock before closing the door, and identifying if that leads to a prohibited state. A door locked permanently open would violate security requirements.

Observations. STAMP models control flows of the system and further analysis identify safety and security hazards in that control flow. Domain experts are required to properly identify issues. STAMP provides a structured way of reasoning based on a high level description of the system that should ensure the identification of safety and security requirements. STAMP can identify safety and security issues in a system, it is not for documenting those interactions. It is suited to help with the syntheses of a model describing safety and security interactions. (**A4**) Paper [32] analyze synchronous-islanded operating microgrids using STAMP.

8.2. SysML

Modelling. The Systems Modeling Language (SysML) is a general-purpose modeling framework for systems engineering, extending the Unified Modeling Language (UML). (**A2**) SysML-sec [76] extends SysML with safety and security requirements. Its functional model describes the communication channels between processes, detailing their encryption methods and the complexity overhead cost. SysML-sec also includes a system mapping model, describing which part of the communication process occurs in which components. SysML-sec enables safety and security properties to be expressed and verified via separate model checkers, e.g., checking if confidential communications can be intercepted by compromising a bus. The interaction between safety and security properties can, however, not be modeled. (**A5**) Since SysML-sec is geared towards embedded software, we replaced the Locked Door Example with a digital keypad lock. Fig. 16 shows the mapping model. Within the keypad, the CPU is connected to the main memory and an Input-Output Bus. Connected to this bus is the physical button pad, as well as a Digital Analog Converter that engages and disengages the lock. Fig. 17 details the process of (dis)engaging the lock. The correct codes are stored in an encrypted format, and an input code is received unencrypted. Both codes are collected, and the input code is checked against the stored key, either by decrypting the stored key or encrypting the input key. On a match, a command is sent to (un)lock the door.

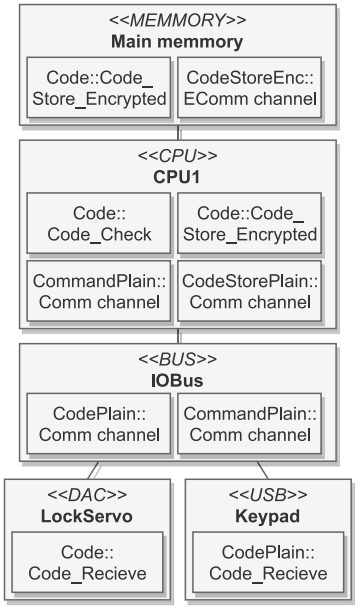

Fig. 16. SysML mapping.

Analysis. (**A3**) The TTool [33] can encode ATs and FTs in a SysML parametric model. It then uses UPPAAL to test for reachability of the TLE. Counter measures can be annotated to the ATs and FTs models, and individual attack vectors/BEs switched on or off. UPPAAL will then indicate if preventing the chosen subset of attack vectors/BEs is sufficient to prevent the TLE. No method for discovering the ATs and FTs from data is included. TTool can also verify security requirements with ProVerif [77]. Requirements can be tagged as e.g. *Confidentiality*, *Non Repudiation*, *Data Origin Authenticity* [78]. A confidentiality requirement on all communications can be created, and communication channels from the functional model are then linked to these requirements. An unsecured bus does not satisfy the confidential requirement, and with no other mitigation, e.g. encrypted messaging, ProVerif will find a state where the confidential requirement of the communication channel is violated, returning a trace showing how this state was reached. Paper [33] uses Key Masters Keying Protocol, which aims to securely distribute a randomly generated key among a group of in-car Electronic Control Units, as a case study.

Observations. (**A1**) In SysML, safety and security are modeled and analyzed separately. Violation of safety and security requirements can be observed, as well as mitigation measures that violate other requirements. (**A5**) SysML does not provide tools for analyzing interactions between requirements, as interactions between safety and security are an emergent behavior of the models.

8.3. ALLOY

Modelling. Alloy [34] is an Object Oriented system modeling language, using set theory to prove assertions on a given model. It consists mainly of *signatures*, which define the structure of classes of objects, *facts*, which define rules the overall system follows, and *assertions*, behaviors that the overall system should comply to, but counterexamples that violate them could exist for. The authors in [79,80] propose (**A2**) Alloy as a basis for creating a system model, and then creating two derivative models, which separately check the safety and security requirements through assertions. This is similar to SysML, which also defines a base behavior of the system for which safety and security are checked separately. (**A4**) The literature we examined contained a case study for a fire safety system [80], but it did not contain a security element.

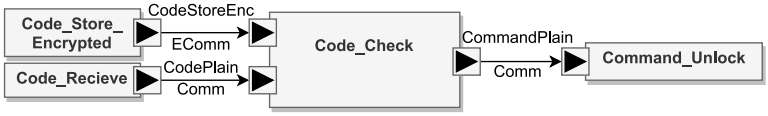

Fig. 17. SysML function diagram for digital door.

Due to a lack of an industrial case study that fits our focus of safety-security co-analysis, we have crafted an Alloy model that captures the locked door example. The first step in defining the model was creating the mechanism for having a Boolean data type. This is achieved by extending an abstract Boolean into a singular *TRUE* and *FALSE*. In stead of requiring a new object for every attribute that could be added to a class, the name of the attribute in combination with pointing to a globally unique *TRUE* and *FALSE* would describe if the property is present or not. Next, *Doors*, *Locks* and the *Building* are defined. A door can be opened or not and a *Lock* engaged or not, state registered by a *Boolean*. A *Door* has a single *Lock* and a *Lock* can be installed in a single *Door*. There is also a *Building* that contains all the *Door*s, which can be on fire described by a *Boolean*. Next we impose some rules on the model to ensure accurate behavior by defining *fact*s. If a *Door* has a *Lock*, then that *Lock* must reflect it is installed in that *Door*, and if a *Lock* identifies a *Door* it is installed in, that *Door* must reflect it has that specific *Lock*. Lastly, all *Door*s that exist must be installed in the *Building* to exclude free floating *Doors*. Three *predicates* are defined, which can be used for evaluation. A *predicate* defining if a specific *Door* d is openable, a *predicate* defining if a person can always escape the *Building* fire, and a *predicate* defining if a burglar is prevented from entering. Lastly, *asserts* are defined, which are used to check if the above defined model fulfills the safety and security requirements. The current model does not fulfill the *asserts*, since the model does not prevent states that violate the *asserts*. However, introducing new *facts* to enforce the safety *assert* either violates the security *assert*, no doors are not openable results in all doors are openable, or requires the *Building* to never be on fire, which is unrealistic.

```
abstract sig Boolean { }
one sig TRUE extends Boolean { }
one sig FALSE extends Boolean{ }
sig Lock {lock_engaged: one Boolean, installed_in: one Door}
sig Door {opened: one Boolean, installed_lock: one Lock}
one sig Building {doors: set Door, on_fire: one Boolean}

fact {all d:Door, o: d.installed_lock | o.installed_in = d}
fact {all l:Lock, o: l.installed_in | o.installed_lock = l}
fact  {all d:Door, b:Building | d in b.doors}

pred  door_openable [d: Door] {d.opened = TRUE or
d.installed_lock.lock_engaged = FALSE}
pred  can_escape_fire {no d: Door, b:Building |
b.on_fire = TRUE and not door_openable[d]
pred  burglar_cant_enter {no d:Door | door_openable[d]}

assert  fire_safe {can_escape_fire}
assert secure {burglar_cant_enter}
assert  safe_and_secure {can_escape_fire
and burglar_cant_enter}
```

Analysis. (**A3**) Alloy is both a language and a toolkit: as such, Alloy models can be automatically analyzed. This is achieved by SAT solving. Alloy can perform checks for a given amount of objects, either a specific amount of objects or all integers up to a specific amount. For these objects, Alloy checks if the assertions hold in all possible configurations of those objects, returning a counterexample where an assertion is violated. A graphical representation of such a counterexample can be automatically generated.

Observations. (**A1**) Alloy can model the absence of interactions by default. Due to the predicate logic, conditional dependency, mutual reinforcement, and antagonism can be encoded directly. Conditional dependency is achieved by defining a mainly safety or security focused predicate that requires a predicate of the other kind to be true. Mutual reinforcement can be modeled by having a safety and a security predicate requiring the same state of another predicate or of a component in the model. Antagonism can be directly modeled by defining a combined safety and security predicate or assert that simultaneously requires a predicate to be true and not true. For example, defining that all doors must simultaneously satisfy and not satisfy the door_openable predicate. Similar to SysML, antagonism can also be detected as an emergent property of the model, where changing the base behavior to satisfy a safety *assert* would violate a security *assert* and vice versa. However we do not count this behavior as explicitly modeling antagonism. While the Alloy language is capable of handling these interactions, using alloy as described in [34] – making separate adaptations of a base model to check safety and security separately – would preclude defining all these interactions. (**A5**) At the time of writing this survey, a new version of Alloy has just released adding support for some temporal relations. No literature exists yet that explores the possibilities of this extension in the context of safety and security. As such, we will consider only the older version of Alloy, and exclude the temporal element in our analysis.

8.4. Event-B

Modelling. Event-B [81–83] has been developed out of the framework RODIN [84]. It is a rigorous approach to correct-by-construction system development. Development starts from the definition of an abstract specification – which models the essential functionalities of the system – that is later refined. In the refinement process, the abstract model is transformed into a detailed specification, expressed using set theory and propositional logic. This propositional logic also supports expressing properties at differing time steps, for example, statements that the difference of variables over time is constrained ($|A(T+1)| < |A(T)| + |\Delta|$). (**A4**) In [35], the authors model the architecture of a battery charging system representing its failure behavior and defining the mechanisms for error detection and recovery. (**A2**) During the refinement process, they also represent the effect of security vulnerabilities such as tampering, spoofing and denial-of-service attacks and analyze their impact on system safety. The step-wise refinement process of Event-B allows one to systematically derive the constraints and define the assumptions that should be fulfilled to guarantee system safety in presence of security attacks. This example is representative of conditional dependency, specifically of security potentially influencing safety. First, we detail the system by defining requirements and constraints. There are two requirements that encode the integrity of the system: 1. You must not be locked in during fire. 2. Attacker must not be able to open the door. There is a single constraint: 1. Door cannot be locked or unlocked in the same time. We assume that door opened = 1 and lock locked = 1, and formalize the state of the system using the variables d and l. We then proceed to refine the model. First refinement: we add guards to the lock and the door events. Second refinement: we add a variable for fire, an invariant and event for fire safety: when fire

happens the door must open, the lock must open to ensure the door can be opened. Third refinement: we add a variable for burglary and an invariant for burglary, plus an event for burglar security. The process of refinement unearths the need for the following property: $f = 0 \vee b = 0$. However, it is not realistic that fire and burglary will never happen simultaneously.

Constants :

Variables : d, l, f, b

Invariants :

Inv01: $d \in \{0, 1\}$

Inv02: $l \in \{0, 1\}$

Inv03: $f \in \{0, 1\}$

Inv04: $b \in \{0, 1\}$

Inv05: $f = 1 \implies d = 1$

Inv06: $b = 1 \implies d = 0$

Properties :

Prop01: $f = 0 \vee b = 0$

Events :

OP – DOOR : **when** $d = 0; l = 0$ **then** $d := 1$

CL – DOOR : **when** $d = 1; l = 0$ **then** $d := 0$

OP – LOCK : **when** $l = 1$ **then** $l := 0$

CL – LOCK : **when** $l = 0$ **then** $l := 1$

FIRE – SAFE : **when** $f = 1$ **then** $l := 0; d := 1$

BURGLAR – SEC : **when** $b = 1$ **then** $l := 1; d := 0$

Analysis. (**A3**) Given some events for the system and its invariants which are the requirements of the system, Event-B can prove properties of that system (e.g., safety-related properties) under given assumptions [85]. However, note that security mechanisms and requirements are still modeled with a similar method as safety ones are, with the same limitations. (**A5**) Event-B is a text-based formalism, thus not allowing graphical visualization of modeled systems.

Observations. (**A1**) As shown, antagonism can be modeled in the lock-door example. Because Event-B is a very general approach – mainly based on set theory and propositional logic – it can model all four kinds of safety and security interactions. In particular, we can encode conditional dependency inside events using guards.

8.5. The architectural analysis and design language

Modelling. The Architectural Analysis and Design Language (AADL) is a framework to model the software and hardware architecture of embedded real-time systems. It is an international standard of the SAE, Aerospace Division [86]. Its core language describes the multi-threaded, distributed software architecture, and the annexes describe real-time behavior and error modeling. The EMV2 language is used to analyze software failures (safety), providing a property set, a library of error models, and an annex sub-language, expressing how errors are generated and propagated. (**A2**) Safety and security are added in one AADL model in [36] as follows: Safety analysis computes the mean time between failures (MTBF) of the system based on the information provided by the AADL model and its error annex. For security, the goal is to avoid unauthorized access to sensitive data. Data is accessed through AADL ports of subprograms. So it is important to verify that the indirect data access points are secure enough. This is done with a dedicated property set that associates a security level (an integer value) to AADL data. Fig. 18 models the Locked Door Example with AADL: this is uncommon for AADL because it is often used for the interaction between software and hardware components. Here we consider a *Person*, a *Locking mechanism*, and *Door access* as components. They are linked through ports. (**A1**) AADL does not model any kind of dependencies: safety and security are considered separately, although they share the same AADL model to perform the analysis.

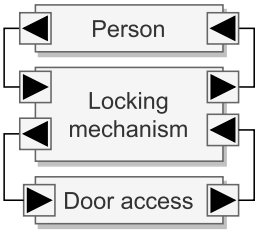

Fig. 18. AADL.

Analysis. (**A3**) For safety, the MTBFs of the system is calculated via the AADL Error model statements in various components descriptions. They are compiled together to generate a FT. For security, the goal is to hide data with help of the Stood for AADL tool.

Observations. An AADL model is scalable and appropriate for version and configuration management, because it can be fully described by its textual representation. AADL can be also integrated into the MILS architectural approach that consists of developing an abstract architecture intended to achieve the stated purpose, and implementing that architecture on a robust technology platform [87]. FTs can be analyzed after transforming AADL to the Arbre Analyste tool, which makes AADL an excellent application for FTs. (**A5**) Interesting for future work is to integrate ATs with AADL. (**A4**) In [36], a toy example is presented, analyzing the safety and security of an electrical locked door. (**A1**) An AADL model does not represent any kind of safety-security dependency, thus it could be valuable to study whether the analysis for classical ATs and FTs combination could help understand if dependencies can be represented via safety-security measures.

9. Expressiveness

Context of comparison. In light of our analysis, we compare the formalisms with respect to their expressiveness, see Fig. 19. We do not consider AADL, SysML or STAMP in terms of expressiveness. While they are powerful system modeling techniques, they do not directly model the safety/security interactions of the system, which is the primary focus of this survey. Only AFTs, BDMPs, BNs, ALLOY and Event-B – although with different methods – fully capture all four interactions: antagonism, mutual reinforcement, conditional dependency and independence. The modeling techniques under analysis are vastly

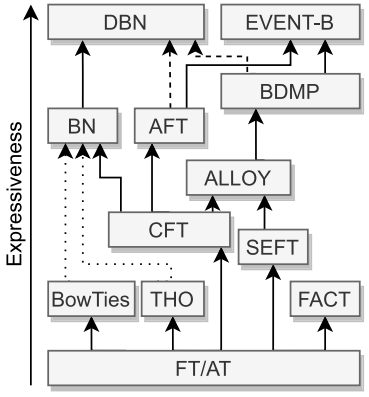

Fig. 19. Expressiveness of formalisms: higher = more expressive. Solid arrows: technique subsumes in features. - - if DBNs support general probability distributions. ⋯ if BNs are expanded to model consequences beyond safety/security TLEs.

different and a one-to-one comparison would not be fair or complete. However, we are not trying to determine an objectively best modeling method. However, despite the difference of considered formalisms – especially when comparing architectural ones and formalisms combining/extending FTs and ATs – we believe a situational comparison can be drawn by focusing on the ability of each formalism to capture safety-security interactions, which is one of the key questions proposed in this paper. Moreover, all the observations regarding key similarities and differences highlighted in each respective section remain valid. Furthermore, we detail the expressiveness analysis of formalisms by considering the following properties: *ability to capture safety-security dependency relations*, *time*, *state of components*, *consequences* (see Table 2).

Expressiveness. FT/AT express independence, directional antagonism, and directional conditional dependency (from security to safety). ATBTs express the same plus consequences after failure. the THO framework mirrors ATBTs to a high degree, but is focused on expected values rather than probabilities. CFT express the same as FT/AT plus bi-directional conditional dependency/antagonism and mutual reinforcement. FACT graphs can model everything that FT/ATs can, but includes the ability to add countermeasures to security or safety risks. SEFTs express the same as FT/AT plus details on the state of components in the system via Petri nets. This enables negation of single components by depending on the non-failed state (an implicit negation), but does not enable negating sub-trees. Thus, SEFTs are slightly less expressive than CFTs (when NOT-gates are included). The attacker steps can also not be influenced by anything in the safety portion, so conditional dependency is directional too. ALLOY's predicate logic can model bidirectional antagonism and negation for entire sub-trees, like CFTs, and does so by exploring the possible states of components, thus is able to detail state like SEFTs, though purely for a qualitative analysis, since it lacks quantitative analysis. The current literature is also not up to date with the state of the ALLOY tool, which now includes the capability to model temporal relationships. When the capabilities for such are properly explored, it might be as expressive as EVENT-B. AFT express the same as CFT, plus an element of time/order through dynamic gates. AFTs use exponential probability distributions, as well as more complex distributions [29]. BNs model all dependencies, multiple TLEs, and potentially can be expanded to model consequences in a method similar to ATBTs, though they have not been used for such in the literature yet. Thus, the introduction of the dotted line in Fig. 19. BDMPs express bidirectional dependencies and details on the state of components via Petri nets, like ALLOY, plus time/order through the use of triggers. Recall they also express probability distributions as mentioned in Section 5.3. Dynamic BN express time, consequences, negation and details on the state of components. They also expand on AFTs and BDMPs if they model generic probability distributions. Event-B, due to its general nature, is the most expressive when it comes to qualitative analysis, though it lacks quantitative analyses. It supports state modeling, and can define system properties over time. Though not shown in literature, it could model consequences such as defined by ATBTs as part of system behavior.

10. Findings and reflection

Our survey revealed several noteworthy findings, some of which are summarized in Tables 2–4.

Finding #1: The majority of approaches combine fault trees and attack trees. Seven of the fourteen formalisms we found combined FTs and ATs. This may not be too much of a surprise, since FTs and ATs are similar in nature and are widely employed, both in industry and academia. Some of these formalisms are plain combinations of FTs and ATs, while some others add constructs to extend modeling/analysis capabilities of these combinations.

Finding #2: No novel modeling constructs are introduced. It is however remarkable that none of the formalisms for safety and security integration that we found introduces novel modeling constructs to capture safety-security interactions. In our opinion, the reason for this phenomenon is to be attributed to familiarity with either safety-specific or security-specific formalisms. In fact, the key strategy that we unearthed is to *merge* already existing safety and security formalisms without adding new operators. Thus, one can represent safety and security features in one model, but one may wonder to what extent the interaction can be expressed appropriately when the main strategy relies on merging existing techniques.

Finding #3: Safety-security interactions are still ill-understood. While the paper [17] coins the four dependency types, these are given in natural language. As such, when one find him/herself trying to apply these definitions to capture interactions via formal methods, it remains unclear how one might translate them formally. In this work, we make a first attempt at proposing rigorous definitions that focus on requirements and events, to then specify these for tree-based formalisms in Section 6. Furthermore, we find that one of the dependencies – i.e., mutual reinforcement – could be better understood by defining reinforcement first, as it appears to not always be a bi-directional interaction. Encouraging more work in translating these definitions for specific categories of formalisms could further narrow this gap. A first practical step might arise from the analysis of real-life case studies, in settings where safety-security interactions are already present.

Finding #4: No novel metrics were proposed. No novel metrics were introduced to quantify safety-security interactions. Again, classical metrics were studied, such as the mean time to failure and attacker success probabilities. These metrics remain anchored to typical safety-specific or security-specific formalisms, and are subsequently merged as modeling constructs are. To address this gap, trade offs between safety and security, e.g., through Pareto analysis, could be studied. Trade offs are – in fact – natural to analyze in these contexts and do not necessitate novel analysis methods.

Table 4
Comparison of case study complexity per analyzed formalism.

Formalism	Metric/Analysis	Size
FT/AT	Probability	8 #Gates
CFTs	Probability	8 #Components
AFTs	Probability, time and cost of attacks	24, 6, 5 #Gates
SEFTs	Probability, Mean Time Between Attacks	6 #Components
FACT Graphs	Probability	9 #Gates
BDMPs	Mean Time to Success, probability of success	19 #Gates
ATBTs	Risk level evaluation, trade offs analysis	12 #Controls
BNs	Mean Time to Failure, conditional independence	16 #Nodes
THO Framework	Likelihood of an attack	/
STAMP	Potential hazards and undesired behaviors	6, 10 #Components
SysML	States that violate properties/requirements	5 #Blocks
ALLOY	Prove properties under given assumptions	18 #Input/Output ports
Event-B	Prove properties under given assumptions	6 #Components in safety goal
AADL	Mean Time Between Failures	9 #Gates (corresp. FT)

Finding #5: No large case studies were carried out. In general, papers analyzed in this work present relatively small examples for illustrative purposes (see Table 4). A notable exception is the medium-size, but realistic, pipeline case study in [19,29]. Another notable exception is the electrical grid case study in [79]. This is remarkable, because for safety and security separately, such numerous large case studies exist, e.g., [88–92]. Conducting a thorough case study analysis in a realistic setting could shed light on how safety and security practically interact, on how one could formally define these interactions in a specific context and could help gather more insight on how safety and security can be jointly analyzed.

Finding #6: diverse formalisms model different safety-security interactions. As per Table 2 and in Section 9, AFTs, BDMPs, BNs, Event-B and ALLOY are the only formalisms that can model all four safety-security dependencies. CFTs and SEFTs can model them when extended/with some limitations, which we suggested in the respective sections, e.g., CFTs need a NOT-gate to express antagonism while SEFTs can only express directional conditional dependency, from security to safety.

11. Conclusion and future work

We now look back at research questions posed in Section 2, offer a summary of answers and pointers to valuable sections in the paper. (**Q1**) How expressive are these formalisms? (**Q2**) Which modeling constructs exist to model the interactions between safety and security? (**Q3**) Which analyses do these formalisms enable? (**Q4**) How do these formalisms compare on industrial case studies and (**Q5**) What are the gaps and the findings? What would be desirable extensions?

In summary:

- (**A1**) Our paper provides a thorough comparison of the analyzed formalisms. For each formalism, we highlight its expressiveness, i.e., their ability to model safety/security interactions, in the appropriate section and we conduct a thorough comparison between them in Section 9. Out of the 14 formalisms analyzed, AFTs, BDMPs, BNs, Event-B and ALLOY are the only 5 formalisms that can model all four safety-security dependencies.
- (**A2**) None of the formalisms present *specific constructs* for safety-security interdependencies: constructs are acquired from safety-only or security-only frameworks and joined afterwards, following different strategies. The same holds for *metrics*: none of those is specific for safety-security interactions. We reflect on reasons behind this phenomenon in Section 10.
- (**A3**) Studied formalisms enable qualitative and/or quantitative analyses: details are provided and highlighted in each section and summarized in Table 3.
- (**A4**) As highlighted in Section 10 and to the best of our knowledge, safety-security interactions are studied inside limited scenarios and large industrial case studies are still missing.
- (**A5**) We tackle gaps in each section and highlight them while analyzing formalisms. Moreover, we summarize our general findings in Section 10. Further extensions could be considered and are suggested in appropriate sections, e.g., for every FT/AT-based formalism, one could employ two roots to account for different safety and security TLEs, as BNs do.

Future work. More rigorous definitions of safety-security dependencies are needed, to account for: 1. *Directionality*. Are safety and security directional or bi-directional and from which direction do they flow? 2. *Intensity*. For a quantifiable co-analysis, intensity of these interaction has to be considered. 3. *Nature of the interaction.* For each of the possible interactions, from reinforcement, to dependency or antagonism, accounting for the positive or negative impact of such an interaction is fundamental. Moreover, conditional dependencies like the one showed in Fig. 4(a) raise the question on who is responsible for depending actions when safety and security are heavily dependent. Game theoretical frameworks could be deployed to analyze this open issue. The visualization of safety/security interdependencies should also be considered to ease readability of complex models. The most recent update of the Alloy tool has introduced more capabilities than explored in the current literature, exploring how the new temporal modeling capabilities can be exploited for safety-security modeling would be of interest as well.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Marielle Stoelinga reports financial support was provided by NWO grant NWA.1160.18.238 (PrimaVera). Marielle Stoelinga reports financial support was provided by EU Marie Sklodowska-Curie grant agreement No 101008233. Stefano Maria Nicoletti, Marielle Stoelinga, Marijn Peppelman reports financial support was provided by ERC Consolidator Grant 864075 (CAESAR). We confirm that this manuscript has not been published elsewhere and is not under consideration by another journal. All authors agree with submission to Computer Science Review. This study was partially funded by the NWO grant NWA.1160.18.238 (PrimaVera), the European Union's Horizon 2020 research and the innovation programme under the Marie Skłodowska-Curie grant agreement No 101008233, and the ERC Consolidator Grant 864075 (CAESAR). The authors have no conflicts of interest to declare.

Data availability

No data was used for the research described in the article.

References

[1] F. Reichenbach, J. Endresen, M.M.R. Chowdhury, J. Rossebø, A pragmatic approach on combined safety and security risk analysis, in: 2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops, 2012, pp. 239–244, http://dx.doi.org/10.1109/ISSREW.2012.98.

[2] C. Woskowski, A pragmatic approach towards safe and secure medical device integration, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2014, pp. 342–353.

[3] A.J. Kornecki, J. Zalewski, Safety and security in industrial control, in: Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, CSIIRW '10, Association for Computing Machinery, New York, NY, USA, 2010, http://dx.doi.org/10.1145/1852666.1852754.

[4] D.P. Eames, J. Moffett, The integration of safety and security requirements, in: International Conference on Computer Safety, Reliability, and Security, Springer, 1999, pp. 468–480.

[5] International Standardization Organization, ISO/DIS 26262: Road Vehicles, Functional Safety, Technical Report, 2009.

[6] ISO/IEC 25010:2011, Systems and software quality requirements and evaluation (SQuaRE), in: System and Software Quality Models, 2011.

[7] A. Avizienis, J.-C. Laprie, B. Randell, C.E. Landwehr, Basic Concepts and Taxonomy of Dependable and Secure Computing, Vol. 1, TDSC, 2004, pp. 11–33.

[8] D.M. Nicol, W. H.Sanders, K.S. Trivedi, Model-based evaluation: From dependability to security, IEEE Trans. Dep. Sec. Comput. 1 (1) (2004) 48–65.

[9] T. Novak, A. Treytl, Functional safety and system security in automation systems - a life cycle model, in: 2008 IEEE International Conference on Emerging Technologies and Factory Automation, 2008, pp. 311–318, http://dx.doi.org/10.1109/ETFA.2008.4638412.

[10] C. Schmittner, Z. Ma, E. Schoitsch, T. Gruber, A case study of FMVEA and CHASSIS as safety and security co-analysis method for automotive cyber-physical systems, in: CPSS, 2015, pp. 69–80.

[11] C. Schmittner, Z. Ma, P. Smith, FMVEA for safety and security analysis of intelligent and cooperative vehicles, in: SAFECOMP, 2014, pp. 282–288.

[12] C. Raspotnig, P. Karpati, V. Katta, A combined process for elicitation and analysis of safety and security requirements, in: Enterprise, Business-Process and Information Systems Modeling, Springer, 2012, pp. 347–361.

[13] K. Lano, D. Clark, K. Androutsopoulos, Safety and security analysis of object-oriented models, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2002, pp. 82–93.

[14] J. Dürrwang, K. Beckers, R. Kriesten, A lightweight threat analysis approach intertwining safety and security for the automotive domain, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2017, pp. 305–319.

[15] G. Macher, A. Höller, H. Sporer, E. Armengaud, C. Kreiner, A combined safety-hazards and security-threat analysis method for automotive systems, in: SAFECOMP, 2014.

[16] A. Mashkoor, F. Kossak, A. Egyed, Evaluating the suitability of state-based formal methods for industrial deployment, Softw. - Pract. Exp. 48 (12) (2018) 2350–2379.

[17] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, Y. Halgand, A survey of approaches combining safety and security for industrial control systems, RESS 139 (2015) 156–178.

[18] M. Sun, S. Mohan, L. Sha, C. Gunter, Addressing safety and security contradictions in cyber-physical systems, in: CPSSW, Citeseer, 2009.

[19] S. Kriaa, M. Bouissou, F. Colin, Y. Halgand, L. Pietre-Cambacedes, Safety and security interactions modeling using the BDMP formalism: case study of a pipeline, in: SAFECOMP, Springer, 2014, pp. 326–341.

[20] F. Arnold, D. Guck, R. Kumar, M. Stoelinga, Sequential and parallel attack tree modelling, in: F. Koornneef, C. van Gulijk (Eds.), Proceedings SAFECOMP Workshops, in: LNCS, vol. 9338, Springer, 2015, pp. 291–299.

[21] B. Kaiser, P. Liggesmeyer, O. Mäckel, A new component concept for fault trees, in: Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software-Volume 33, Citeseer, 2003, pp. 37–46.

[22] M. Steiner, P. Liggesmeyer, Combination of safety and security analysis - finding security problems that threaten the safety of a system, in: SAFECOMP, 2016.

[23] I.N. Fovino, M. Masera, A. De Cian, Integrating cyber attacks within fault trees, Reliab. Eng. Syst. Saf. 94 (9) (2009) 1394–1402.

[24] M. Bouissou, J.-L. Bon, A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes, Reliab. Eng. Syst. Saf. 82 (2) (2003) 149–163.

[25] H. Abdo, M. Kaouk, J.-M. Flaus, F. Masse, A safety/security risk analysis approach of industrial control systems: A cyber bowtie–combining new version of attack tree with bowtie analysis, Comput. Secur. 72 (2018) 175–195.

[26] H. Abdo, M. Kaouk, J.-M. Flaus, F. Masse, A new approach that considers cyber security within industrial risk analysis using a cyber bow-tie analysis, 2017, https://hal.archives-ouvertes.fr/hal-01521762, working paper or preprint.

[27] G. Sabaliauskaite, A.P. Mathur, Aligning cyber-physical system safety and security, in: Complex Systems Design & Management Asia, Springer, 2015, pp. 41–53.

[28] M. Roth, P. Liggesmeyer, Modeling and analysis of safety-critical cyber physical systems using state/event fault trees, in: SAFECOMP, 2013.

[29] R. Kumar, M. Stoelinga, Quantitative security and safety analysis with Attack-Fault Trees, in: 18th International Symposium on HASE, 2017, pp. 25–32.

[30] A.J. Kornecki, N. Subramanian, J. Zalewski, Studying interrelationships of safety and security for software assurance in cyber-physical systems: Approach based on bayesian belief networks, in: 2013 FEDCSIS, IEEE, 2013, pp. 1393–1399.

[31] T. Aven, A unified framework for risk and vulnerability analysis covering both safety and security, Reliab. Eng. Syst. Saf. 92 (6) (2007) 745–754.

[32] I. Friedberg, K. McLaughlin, P. Smith, D. Laverty, S. Sezer, STPA-SafeSec: Safety and security analysis for cyber-physical systems, J. Inf. Secur. Appl. 34 (2017) 183–196.

[33] G. Pedroza, L. Apvrille, D. Knorreck, AVATAR: A SysML environment for the formal verification of safety and security properties, in: NOTERE, 2011, pp. 1–10.

[34] D. Jackson, Alloy: a lightweight object modelling notation, ACM Trans. Softw. Eng. Methodol. (TOSEM) 11 (2) (2002) 256–290.

[35] I. Vistbakka, E. Troubitsyna, T. Kuismin, T. Latvala, Co-engineering safety and security in industrial control systems: a formal outlook, in: International Workshop on Software Engineering for Resilient Systems, Springer, 2017, pp. 96–114.

[36] P. Dissaux, F. Singhoff, L. Lemarchand, H. Tran, I. Atchadam, Combined real-time, safety and security model analysis, in: ERTSS, 2020.

[37] D.S. Velasco Moncada, Hazard-driven realization views for component fault trees, Softw. Syst. Model. 19 (6) (2020).

[38] A. Mashkoor, A. Egyed, R. Wille, Model-driven engineering of safety and security systems: A systematic mapping study, 2020, arXiv preprint arXiv:2004.08471.

[39] S. Chockalingam, D. Hadžiosmanović, W. Pieters, A. Teixeira, P. van Gelder, Integrated safety and security risk assessment methods: A survey of key characteristics and applications, Lect. Not. Comput. Sci. 10242 (2017) 50–62.

[40] V. Nigam, A. Pretschner, H. Ruess, Model-based safety and security engineering, 2019.

[41] A. Mashkoor, A. Egyed, R. Wille, Model-driven engineering of safety and security systems: A systematic mapping study, 2020, CoRR abs/2004.08471. https://arxiv.org/abs/2004.08471.

[42] C. Raspotnig, A.L. Opdahl, Comparing risk identification techniques for safety and security requirements, J. Syst. Softw. 86 (4) (2013) 1124–1151, http://dx.doi.org/10.1016/j.jss.2012.12.002.

[43] J. Brocke, A. Simons, B. Niehaves, K. Riemer, R. Plattfaut, A. Cleven, Reconstructing the giant: On the importance of rigour in documenting the literature search process, in: ECIS, 2009.

[44] E. Lisova, I. Šljivo, A. Čaušević, Safety and security co-analyses: A systematic literature review, IEEE Syst. J. 13 (3) (2018) 2189–2200.

[45] S. Bhunia, M.S. Hsiao, M. Banga, S. Narasimhan, Hardware trojan attacks: Threat analysis and countermeasures, Proc. IEEE 102 (8) (2014) 1229–1247.

[46] Microsoft Security Development Lifecycle, Threat modeling, 2021, https://microsoft.com/en-us/securityengineering/sdl/threatmodeling, [Accessed 29 June 2021].

[47] S. Kriaa, M. Bouissou, Y. Laarouchi, A model based approach for SCADA safety and security joint modelling: S-cube, 2015.

[48] S. Kriaa, M. Bouissou, Y. Laarouchi, SCADA safety and security joint modeling (s-cube): case study of a dam, in: Proceedings of the 22th Computer & Electronics Security Applications Rendez-Vous (C&ESAR'2015), 2015, pp. 55–69.

[49] Z.B. Celik, P. McDaniel, G. Tan, Soteria: Automated iot safety and security analysis, in: 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18), 2018, pp. 147–158.

[50] W.G. Temple, Y. Wu, B. Chen, Z. Kalbarczyk, Systems-theoretic likelihood and severity analysis for safety and security co-engineering, in: International Conference on Reliability, Safety and Security of Railway Systems, Springer, 2017, pp. 51–67.

[51] International Electrotechnical Commission, IEC 61025: Fault Tree Analysis (FTA), IEC Standards Online, 2006.

[52] C.A. Ericson, Fault tree analysis, in: Sys. Safety Conf., Vol. 1, 1999, pp. 1–9.

[53] E. Ruijters, M. Stoelinga, Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools, Comp. Sci. Rev. 15–16 (2015) 29–62.

[54] B. Schneier, Modeling security threats, Dr. Dobb's J. 24 (12) (1999).

[55] B. Kordy, L. Pièvre-Cambacédès, P. Schweitzer, DAG-based attack and defense modeling: Don't miss the forest for the attack trees, Comput. Sci. Rev. 13–14 (2014) 1–38.

[56] E.J. Zampino, Application of fault-tree analysis to troubleshooting the NASA GRC icing research tunnel, in: RAMS, 2001, pp. 16–22.

[57] M. Fraile, M. Ford, O. Gadyatskaya, R. Kumar, M. Stoelinga, R. Trujillo-Rasua, Using attack-defense trees to analyze threats and countermeasures in an ATM: a case study, in: IFIP, Vol. 267, Springer, 2016.

[58] E.J. Byres, M. Franz, D. Miller, The use of attack trees in assessing vulnerabilities in SCADA systems, in: Int. Infrastructure Survivability Workshop, 2004, pp. 3–10.

[59] C. Budde, C. Kolb, M. Stoelinga, Attack trees vs. Fault trees: Two sides of the same coin from different currencies, 2021, pp. 457–467.

[60] E. Ruijters, D. Guck, P. Drolenga, M. Stoelinga, Fault maintenance trees: Reliability centered maintenance via statistical model checking, in: RAMS, 2016.

[61] S. Junges, D. Guck, J. Katoen, M. Stoelinga, Uncovering dynamic fault trees, in: 2016 DSN, 2016, pp. 299–310.

[62] J.B. Dugan, S.J. Bavuso, M.A. Boyd, Dynamic fault-tree models for fault-tolerant computer systems, IEEE Trans. Reliab. 41 (3) (1992) 363–377.

[63] B. Kordy, S. Mauw, S. Radomirović, P. Schweitzer, Attack–defense trees, LOGCOM 24 (1) (2012) 55–87.

[64] S. Karnouskos, Stuxnet worm impact on industrial cyber-physical system security, in: IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2011, pp. 4490–4494.

[65] M. Bouissou, Automated Dependability Analysis of Complex Systems with the KB3 Workbench: the Experience of EDF R&D, CIEM, 2005.

[66] C.E. Budde, P.R. D'Argenio, R.E. Monti, Compositional construction of importance functions in fully automated importance splitting, in: VALUETOOLS, 2016.

[67] D.S. Nielsen, The Cause/consequence Diagram Method as a Basis for Quantitative Accident Analysis, Tech. Rep., Danish Atomic Energy Commission, 1971.

[68] L. Arnaboldi, D. Aspinall, Towards interdependent safety security assessments using bowties, in: M. Trapp, E. Schoitsch, J. Guiochet, F. Bitsch (Eds.), Computer Safety, Reliability, and Security. SAFECOMP 2022 Workshops - DECSoS, DepDevOps, SASSUR, SENSEI, USDAI, and WAISE, Munich, Germany, September 6-9, 2022, Proceedings, in: Lecture Notes in Computer Science, vol. 13415, Springer, 2022, pp. 211–229, http://dx.doi.org/10.1007/978-3-031-14862-0_16.

[69] M. Stoelinga, C. Kolb, S.M. Nicoletti, C.E. Budde, E.M. Hahn, The marriage between safety and cybersecurity: Still practicing, in: A. Laarman, A. Sokolova (Eds.), Model Checking Software - 27th International Symposium, SPIN 2021, Virtual Event, July 12, 2021, Proceedings, in: Lecture Notes in Computer Science, Vol. 12864, Springer, 2021, pp. 3–21, http://dx.doi.org/10.1007/978-3-030-84629-9_1.

[70] S. Montani, L. Portinale, A. Bobbio, Dynamic Bayesian networks for modeling advanced fault tree features in dependability analysis, in: Proceedings of the Sixteenth European Conference on Safety and Reliability, 2005, pp. 1415–1422.

[71] N. Leveson, A new accident model for engineering safer systems, Saf. Sci. 42 (4) (2004) 237–270.

[72] C. Schmittner, Z. Ma, P. Puschner, Limitation and improvement of STPA-sec for safety and security co-analysis, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2016, pp. 195–209.

[73] S. Procter, E.Y. Vasserman, J. Hatcliff, SAFE and secure: Deeply integrating security in a new hazard analysis, in: Proceedings of the 12th International Conference on Availability, Reliability and Security, 2017, pp. 1–10.

[74] D. Pereira, C. Hirata, R. Pagliares, S. Nadjm-Tehrani, Towards combined safety and security constraints analysis, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2017, pp. 70–80.
[75] W. Young, N.G. Leveson, An integrated approach to safety and security based on systems theory, Commun. ACM 57 (2) (2014) 31–35, http://dx.doi.org/10.1145/2556938.
[76] Y. Roudier, L. Apvrille, SysML-Sec: A model driven approach for designing safe and secure systems, in: MODELSWARD, IEEE, 2015, pp. 655–664.
[77] R. Ameur-Boulifa, F. Lugou, L. Apvrille, SysML model transformation for safety and security analysis, in: CSITS, Springer, 2018, pp. 35–49.
[78] L. Apvrille, TTool: SysML-Sec Tutorial, Sophia-Antipolis, France, 2020, at https://ttool.telecom-paris.fr/docs/sysmlsec_documentation.pdf.
[79] J. Brunel, D. Chemouil, L. Rioux, M. Bakkali, F. Vallée, A viewpoint-based approach for formal safety & security assessment of system architectures, in: 11th Workshop on Model-Driven Engineering, Verification and Validation, Vol. 1235, 2014, pp. 39–48.
[80] J. Brunel, D. Chemouil, Safety and security assessment of behavioral properties using alloy, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2014, pp. 251–263.
[81] J.-R. Abrial, Event driven system construction, Rapport Tech. Clearsy 15 (1999).
[82] J.-R. Abrial, Modeling in Event-B: System and Software Engineering, Cambridge University Press, 2010.
[83] D. Cansell, D. Méry, Event B, in: H. Habrias, M. Frappier (Eds.), Software Specification Methods, HERMES, 2006, https://hal.inria.fr/inria-00096696.
[84] C. Snook, M. Butler, A. Edmunds, I. Johnson, Rigorous development of reusable, domain-specific components, for complex applications, 2004.
[85] E. Troubitsyna, L. Laibinis, I. Pereverzeva, T. Kuismin, D. Ilic, T. Latvala, Towards security-explicit formal modelling of safety-critical systems, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2016, pp. 213–225.
[86] AADL, AADL: Architecture analysis and design language, 2021, http://www.aadl.info, [Accessed 1 July 2021].
[87] A. Cimatti, R. DeLong, D. Marcantonio, S. Tonetta, Combining MILS with contract-based design for safety and security requirements, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2014, pp. 264–276.
[88] M. Bozzano, A. Cimatti, J.-P. Katoen, V.Y. Nguyen, T. Noll, M. Roveri, Safety, dependability and performance analysis of extended AADL models, Comput. J. 54 (5) (2011) 754–775.
[89] C. von Essen, D. Giannakopoulou, Analyzing the next generation airborne collision avoidance system, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2014, pp. 620–635.
[90] M. Bozzano, A. Cimatti, M. Gario, D. Jones, C. Mattarei, Model-based safety assessment of a triple modular generator with xSAP, Formal Aspects Comput. 33 (2) (2021) 251–295.
[91] M. Fraile, M. Ford, O. Gadyatskaya, R. Kumar, M. Stoelinga, R. Trujillo-Rasua, Using attack-defense trees to analyze threats and countermeasures in an ATM: a case study, in: IFIP Working Conference on the Practice of Enterprise Modeling, Springer, 2016, pp. 326–334.
[92] S. Baloglu, S. Bursuc, S. Mauw, J. Pang, Election verifiability revisited: Automated security proofs and attacks on helios and belenios, in: 2021 IEEE 34th Computer Security Foundations Symposium (CSF), IEEE, 2021, pp. 1–15.