Journal of Information Security and Applications 47 (2019) 302–319

Contents lists available at ScienceDirect

Journal of Information Security and Applications

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

A STAMP-based ontology approach to support safety and security analyses

Daniel Patrick Pereira[a,*], Celso Hirata[a], Simin Nadjm-Tehrani[b]

[a] Department of Computer Science, ITA Instituto Tecnologico de Aeronautica, Sao Jose dos Campos, Brazil
[b] Department of Computer and Information Science, Linköping University, Linköping, Sweden

ARTICLE INFO

Keywords:
Safety assessment
Security assessment
Security ontology
STAMP/STPA-Sec

ABSTRACT

Considerations of safety and security in the early stage of system life cycle are essential to collect and prioritize operation needs, determine feasibility of the desired system, and identify technology gaps. Experts from many disciplines are needed to perform the safety and security analyses, ensuring that a system has the necessary attributes. Safety assessment is usually conducted in the concept stage. On the order hand, security assessment is performed in design stage usually when an initial architecture along with the logical and physical components are defined. Systems-Theoretic Process Analysis (STPA) is a new hazard analysis technique based on systems thinking and is built on top of a new causality model of accident, which stands for Systems-Theoretic Accident Model and Processes (STAMP), grounded in systems theory. STPA for Security (STPA-Sec) is an extension of STPA that proposes to include security concerns into the analysis. STPA-Sec helps identifying some hazardous control actions, causal scenarios, and casual factors; however, no emphasis is placed on security threat scenarios. In this paper we propose an ontology-based technique that extends STPA-Sec to improve identification of causal scenarios and associated casual factors, specifically those related to security. We propose an approach that assists safety and security experts conducting safety and security analyses using STPA-Sec with a supporting ontology. First, we present an ontology representing the safety and security knowledge through STPA-Sec process, and provide a tool that implements the proposed ontology. We then propose a process to capture safety and security knowledge into the proposed ontology to identify causal scenarios. We perform a preliminary evaluation of the ontology and the process using an aeronautic case study. The results show that the ontology-based approach helps systems engineers to identify more security scenarios compared to the case where they use only STPA-Sec. Furthermore, some hazardous control actions are not addressed if the systems engineer uses the basic STPA-Sec.

© 2019 Elsevier Ltd. All rights reserved.

1. Introduction

The concept stage is where the stakeholder needs are closely examined, defining functionalities, and multidisciplinary analysis (e.g., safety and security) is conducted. In general, many projects succumb by abbreviating the conceptual stage. This means that the systems engineer has short time to perform the analysis of concerns in a comprehensive manner including those of safety and security. Alternatively, he/she performs the analysis somewhat late in the development cycle, when many compromises are already made, restricting the design to meet specific concerns.

Systems engineering is an interdisciplinary approach that manages the complete development of systems, from customer needs through operations to retirement. The concept stage requires experts of many areas to perform trade-off analyses, help making decisions, and arrive at a suitable solution. For the systems we are building today, safety and security analyses at the early stage of the system life cycle are essential to collect and prioritize operational needs, determine the feasibility of the desired system, and identify technological gaps.

The safety discipline has matured over decades in the aerospace. SAE ARP-4754A and ARP-4761 provide a guidance for development of civil aircraft and systems, and safety assessment process on civil airborne systems and equipment, respectively. On the other hand, security discipline is quite young, and some industries have been defining their own standards. Thus, EUROCAE / RTCA recently has provided guidance to cover security requirements (ED-202A / DO-326A and ED-203A / DO-356A) that are related to the system functions. Therefore, the security engineering is becoming an integral part of the systems engineering process in aviation industry.

* Corresponding author.
E-mail address: dpatricksp@gmail.com (D.P. Pereira).

https://doi.org/10.1016/j.jisa.2019.05.014
2214-2126/© 2019 Elsevier Ltd. All rights reserved.

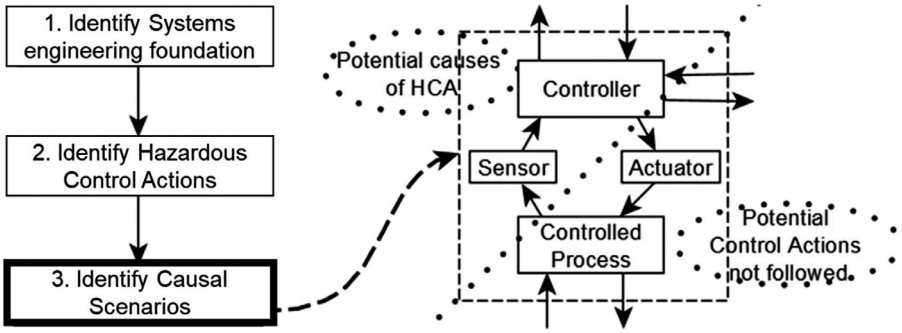

Fig. 1. STPA-Sec process.

The safety assessment is usually performed in the concept stage. However, security assessment is accomplished in more advanced stages, usually when an initial architecture along with the logical and physical components are defined. A proper multi-disciplinary analysis made in the early stages of system life cycle might avoid recalls and rework. Our thesis is that the concept stage should provide information to define the functional system architecture and use that to analyze safety and security concerns.

In Information Science, an ontology encompasses a representation, formal naming, and definition of the categories, properties, and relations between the concepts, data, and entities that substantiate one, many, or all domains. Every domain creates ontologies to limit complexity and organize information into data and knowledge.

There are several safety and security ontologies. Safety ontologies relate to identification of hazards, and losses, while security ontologies explore categories such us vulnerabilities, attacks, and data breaches or service disruption. To the best of our knowledge, there is no ontology that covers safety and security jointly. A unified ontology allows identifying commonalities and relations and allows a more comprehensive analysis.

Safety and security experts' engagement in advanced system life cycle stages might lead to system redesign with a high cost, which can affect the system schedule. Similar to safety analysis, security analysis requires the systems engineer to identify specific types of security scenarios. The identification of security scenarios requires security expertise from the systems engineer. In addition, the identification of security scenarios can take long time in case the systems engineer is not familiar with security aspects.

STPA-Sec is an extension of STPA (including safety analysis) for security (Young and Leveson [14]). STPA-Sec enables analysis of security as a high-level strategic problem rather than a tactical problem. Safety and security concerns can be addressed together at early stages of system life cycle. Fig. 1 shows the basics STPA-Sec activities. The first activity *Identify Systems Engineering foundation* identifies the mission, unacceptable losses, hazards, and the functional control structure. The second activity, known as Step 1, identifies the *Hazardous Control Actions* (HCA), which include unsafe and unsecure control actions. The third activity, known as Step2, which is the focus of this work identifies the reasons why hazardous control actions might occur.

The right side of Fig. 1 is the control loop used to identify the potential HCA. It is divided in two diamonds; the superior part identifies when control actions might be issued causing hazardous states and the inferior part when control actions are not followed leading to a hazardous state.

STPA-Sec does not emphasize the analysis of security scenarios. STPA-Sec is proposed based on STPA, which primarily considers non-intentional causes. We hypothesize that the combining ontologies can contribute with STPA-Sec will improve identification of causal scenarios and associated casual factors, emphasizing those related to security threats. The goal of this work is to support the safety and security engineers identify the mitigations need to address the identified hazards using STAMP-based approach. Hence, we need to strengthen their understanding of the causal scenarios and causal factors that lead to those hazards.

The contribution of this paper are: (i) define an ontology-based approach, which consists of a tool to retrieve the store knowledge on security elements obtained in earlier and analysis and assist the systems engineer to conduct the analysis; (ii) develop the STAMP-based ontology tool (SOT) that implements the proposed ontology and provides the causal factors, attacks, and recommendations; and (iii) create a process to describe how to feed and retrieve information from the knowledge repository. In particular, we support the identification of relevant scenarios using known attack mechanisms, and propose known mitigation mechanisms. Reusing common parts of previous safety and security analyses allows safety and security experts dedicate more attention to the new functionalities added to the system.

The rest of this paper is organized as follow. In Section 2, we describe the related work, highlighting the gaps and similarities with our work. Section 3 presents the proposed STAMP-based ontology, STAMP-based ontology tool SOT, and the proposed process. A case study is presented in Section 4. Section 5 presents the evaluation of the approach. Section 6 shows the discussions, conclusions and future work.

2. Related work

Rosa et al. [9] present a literature survey on ontologies and taxonomies concerning the security assessment domain. A systematic review approach in seven scientific databases was conducted where 138 papers were identified and divided into categories according to their main contributions, namely: ontology, taxonomy and survey. The selected papers are divided into categories such as main contribution, research issue, application domain, and characteristics. Authors identify the gaps in research issues such as reusing knowledge, automating processes, increasing coverage of assessment, secure sharing of information, defining security standards, identifying vulnerabilities, measuring security, protecting assets, and assessing, verifying or testing the security. Our proposed ontology copes with reusing knowledge, automating processes, and measuring security, protecting assets, and identifying vulnerabilities.

Rosa et al. [10] propose an ontology, named Security Assessment Ontology (SecAOnto), to deal with security assessment aspects and particularities. The authors point out the misunderstanding between information security and systems assessment areas. SecAOnto is based on glossaries, vocabularies, taxonomies, ontologies, and market's guidelines. The SecAOnto includes concepts related to systems assessment, information security, and security assessment; it builds on several entities in earlier security

ontologies (e.g., [7]). The ontology uses an algorithm for calculating coverage of assessment dimensions and security properties. The Assessment Dimension (e.g., Business Logic, System Architecture, Process, and System in Runtime) entity defines under which view or perspective the system must be evaluated. SecAOnto does not intend to deal with risk management process, nor vulnerability analysis, while our proposed covers them.

The most recent work similar to our approach is presented by Souag et al. [11]. They implement the ontology and develop an interactive environment to facilitate the use of the ontology during the security requirements engineering process. The proposed core ontology is organized in three dimensions namely organization, risk, and treatment. The requirement engineer selects the valuable asset along with a location, and then the threats are listed. From the threats, the vulnerabilities are identified, and security requirements are generated. The paper does not describe the process in place to feed the proposed ontology. Different from our approach, the stored knowledge in the Souag et al. [11] ontology is about an already designed system, while our proposed approach is meant to be employed in the concept stage.

Vasilevskaya [13] proposes the Security-Enhanced Embedded system Design (SEED) that is an approach to support the development of security-enhanced systems. SEED supports transferring of knowledge from security experts to embedded systems engineers. With this thought, two ontologies were developed; the first one, named Domain-Specific Security Model (DSSM), is used by a security expert to describe a security solution, and the second one, named Performance Evaluation Record (PER), describes results of performance evaluation of a security mechanism, hence qualifying its resource footprint. The embedded systems engineers can use DSSM, through a developed tool, to select the asset, and then from the security ontology, a set of security properties associated with the identified assets is presented. Our work is related in the sense that we intend to reduce the workload of the specialists. Vasilevskaya [13] focuses on resolving security issues at the design phase whereas we focus on covering safety and security issues at the concept stage.

Heerden et al. [5] present the network attack ontology to classify a wide range of computer network attacks. The attack scenario class is subdivided into ten types of network attacks, which provides the means to classify the different attacks. Motivation, attach mechanism, actor location are some examples used to classify attacks. The authors do not demonstrate how to use the proposed ontology, neither a process to feed it. Although the network attack ontology focuses on physical attacks, we see that the proposed ontology can be adapted to fit our proposed STAMP-based ontology.

Massacci et al. [8] propose a unified security ontology based on an agent-oriented software (SI*), and anti-requirements analysis (Abuse Frames). The extended ontology approach is applied in the ADS-B system to model a security requirement problem domain at different levels of abstraction. A threat scenario is modeled to show the spoofing of the GPS signal; it shows the relationships between the resources, assets, anti-goals, actor, and attacker. The paper does not describe the process in place to use the extended ontology in a complex system context, including safety-criticality aspects. The modeled ASD-B example does not consider other elements in the scenario such as cockpit crew for instance.

Elahi et al. [3] propose a vulnerability-centric modeling ontology, which aims to integrate empirical knowledge of vulnerabilities into the security requirements conceptual foundations. The proposed ontology aims to detect the missing security constructs in security requirements modeling frameworks and facilitate their enhancement. The authors revise the CORAS risk modeling by adopting the proposed ontology. The proposed ontology does not decompose some classes such as Vulnerability and Attack, while Herzog et al. [7] explore the knowledge for creating subclasses. The Effect class is characterized by the attribute severity; however, it is not clear how it is applied. In our proposed ontology we try to cover severity with Severity class, and add other class named Level of Threat to evaluate the likelihood of a successful attack.

Goluch et al. [4] propose to combine the ROPE (Risk-Oriented Process Evaluation) methodology and the security ontology to enable risk-aware business process management and optimization. The ROPE methodology consists of five iterative processes derived from the BPMS (Business Processing Modeling Systems) paradigm. The security ontology is populated carrying out the ROPE methodology; performing a simple web service query is possible to extract the threatened infrastructure. In order to validate, the authors develop a proof of concept prototype, implemented by toolset and claim the feasibility of the approach and the added value gained through the combination of the security ontology and ROPE methodology.

Herzog et al. [7] present a public and available OWL-based ontology of information security, which models assets, threats, vulnerabilities, countermeasures and their relations. The core ontology is able to answer queries, and supports machine reasoning. Some amount of data is available for consulting; the available data is generated based on classic components of risk analysis, which contains information regarding SSH, and buffer overflow for instance. Most of the classes in this ontology are tied to the physical architecture (e.g., SSH, VPN).

Zhou et al. [15] present the Hazard Ontology (HO), which serves as an ontological interpretation of hazards, together with real-world semantics. HO stores three kinds of facts about hazards, in the sense of (i) concepts that represent entities of importance to interpret the concept of hazard and, (ii) relations that are labeled and directed connections between concepts and, (iii) axioms that are used to model knowledge that is always true. The authors present a set of questions to assist the analyst to categorize the preliminary hazard analysis using the HO.

Zhou et al. [16] also propose an approach, based on the HO, to explore and identify the causes associated with the hazards from a Preliminary Hazard Analysis (PHA), aiming to improve the identification of the causes of hazards in terms of completeness and usefulness. The proposed ontology identifies the causes associated with the hazard descriptions from a PHA worksheet. The authors argue that the approach improves the results of hazard causes identification. Our work differs from theirs by providing support for both the safety and security analysis and their relationships.

Bloomfield and Vargas [1] propose a mechanized method that can improve the analysis of results obtained at the early stages of system design. From preliminary hazard worksheets, the authors use an ontology to check consistency and well-formedness of information contained in the preliminary documents. The Object Process Methodology (OPM) is used to model the system from an ontology perspective. They present the reasoning approach for direct and indirect causality. Our work is related in the sense that we have in our ontology hazard, and cause entities; however we make use of scenarios (e.g., safety and security) to explore, and discover new causal factors.

Ebrahimipour and Yacout [2] present an ontology-based FMEA. The ontology-based FMEA entities are component, failure mode, probability of occurrence, probability of detection, risk value, and so on. Generic failure modes are identified, and can be used during the analysis. New failure modes can be added for specific analysis. The focus is on the component level when there is a physical architecture, while our approach is employed at a higher abstraction layer.

3. Proposed approach

This section describes the elements of STAMP-based Ontology along with the general relationship among them. The proposed process to employ the ontology is also presented.

3.1. STAMP-based ontology description

The proposed STAMP-based Ontology is based on existing safety and security ontologies. We aim to create a unified safety and security ontology for using the STPA-Sec process in order to carry out safety and security analyses. Hence, some new entities exclusive to STAMP are added.

We conceptualize the STAMP-based ontology to combine safety and security disciplines to support the systems engineer to conduct the analysis in the concept stage. A unified ontology enables identifying and understanding how safety and security might violate the critical properties of the system under analysis.

STPA-Sec Step 2 employs guidewords to assist the systems engineer to generate the security scenarios. However, our experience shows that the systems engineer needs further information to generate security scenario and recommendations, which is not presented by the STPA-Sec approach. The information is related to attack mechanisms, which enables generating the security scenarios in presence of intentional failures and abuse. Therefore, the unified ontology enhances the knowledge applied in different paths to create the security scenarios. Besides, the security recommendation characterized within the ontology will support the systems engineer in creating security constraints.

The proposed unified STAMP-based Ontology is shown in Fig. 2. It illustrates an overview of the ontology, showing how its entities relate to each other. Some entities and relations are omitted and are explained in the upcoming figure. The rectangles filled with yellow, labeled by (1), represent entities found in safety ontologies, such as one presented by Zhou et al. [15,16], and they have the same meaning for STAMP. The rectangles filled with blue labeled by (2), represent entities of security ontologies ([7,9]), and the white filled rectangles represent entities introduced by this work. All the entities are explained in what follows.

Mission comprises purposes, methods, and goals to carry out specific objectives. It describes what the system is supposed to do, defining and framing the problem. This is the starting point for conducting STPA-Sec analysis. *Purpose* states what the system must do, *Method* elicits the key activities necessary to conduct the mission (i.e. how to do it), and then why the system must do it is goal.

UnacceptableLoss or *Mishap* (named by Zhou et al. [15,16]) is the loss of something valuable and/or that is not acceptable by stakeholders. Losses may generally include loss of human life, human injury, property damage, environmental damage, loss of mission, loss of reputation, and leaking of sensitive information. The *UnacceptableLoss* comes from STAMP (Leveson [20]) and *Mishap* from (Zhou et al. [15,16]). Both terms are equivalent and used interchangeably here. The entity *Hazard* has the same meaning by Leveson [20] and Zhou et al. [15,16]. A *Hazard* is a system state or set of conditions that, together with a particular set of worst-case environmental conditions can lead to an *UnacceptableLoss*.

HazardousControlAction is a control action provided by a *STPASecElement* (issued by a controller; the controller and the issuance are not shown in Fig. 2) that, in a particular context and worst-case environment conditions, can lead to a *Hazard*. In STPA-Sec, *HazardousControlAction* is an unsafe or an unsecure control action. *STPASecElement* is explained later in Fig. 3.

The entity *CausalScenario* relates to the entity *Severity*, which is a qualitative indication of the magnitude of the adverse effect of a *CausalScenario*. The entity *Severity* is evaluated in the same manner as a Functional Hazard Analysis, it usually comes from the safety team, and can have the following values: Catastrophic, Hazardous, Major, Minor, and No Effect. The entities *SafetyScenario*

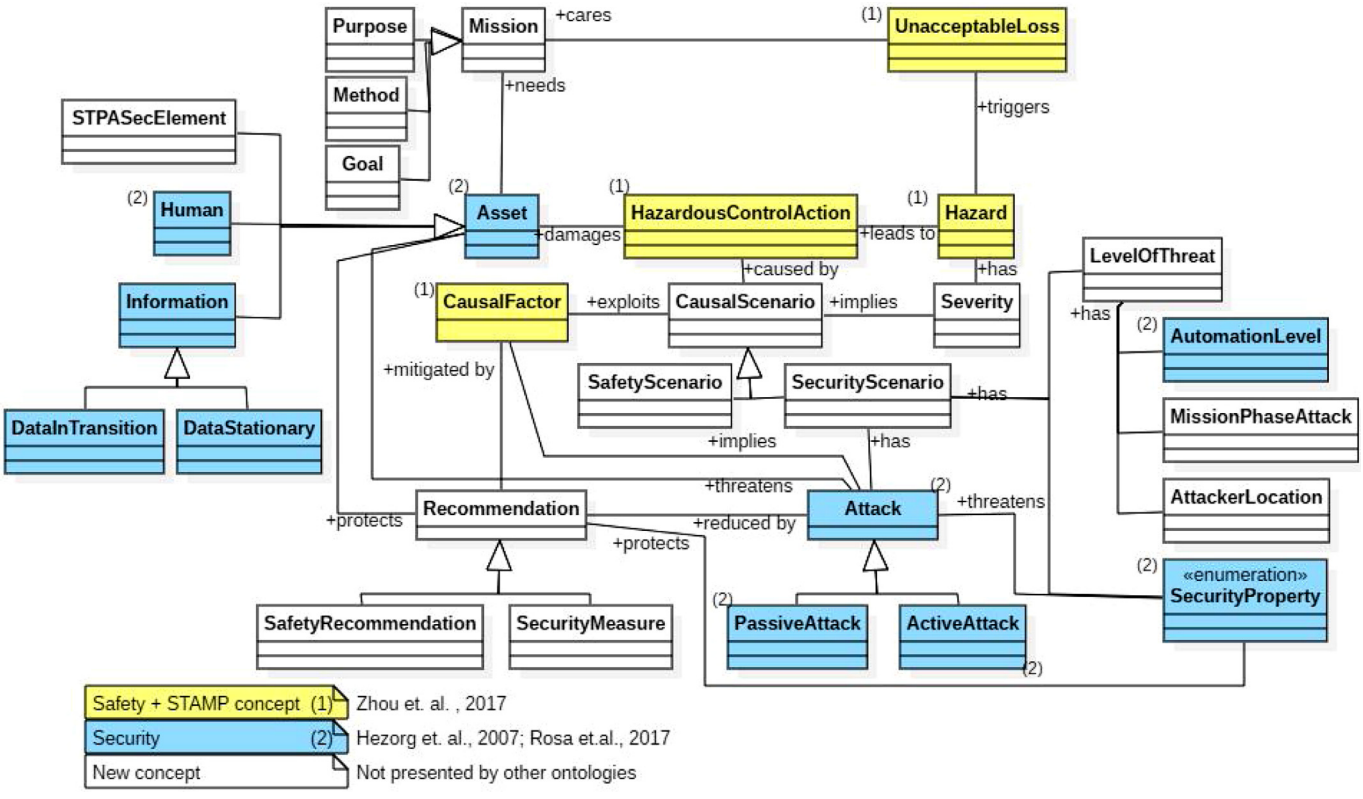

Fig. 2. STAMP-based ontology (For interpretation of the references to color in this figure, the reader is referred to the web version of this article).

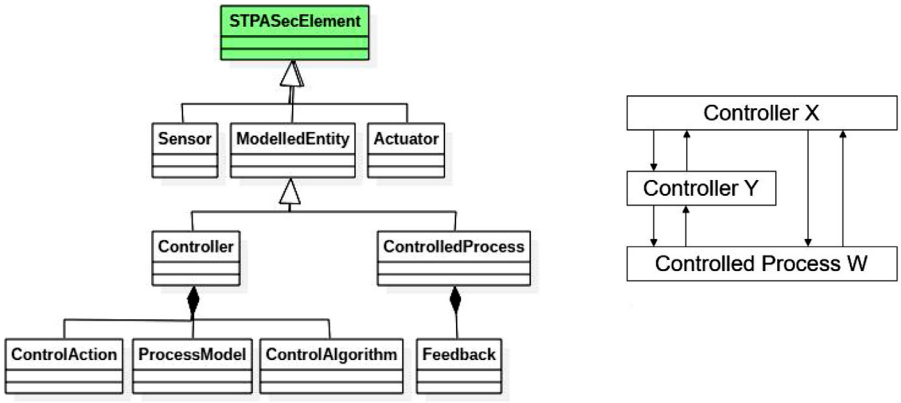

Fig. 3. Entity STPA-Sec element.

and *SecurityScenario* are subclasses of the entity *CausalScenario*. *SafetyScenario* covers the unintentional actions that describe how incorrect feedback, design errors, component failures, and other factors can lead to a *HazardousControlAction* and *UnacceptableLoss*. Additionally, *SecurityScenario* covers intentional actions, explaining how a control flaw can be introduced by an adversary. The *SecurityScenario* is our means to reveal unknown scenarios that STPA-Sec does not cover with the guidewords.

STPA-Sec emphasizes the identification of causal factors in order to provide an explanation of why an unacceptable loss occurs. By utilizing this explanation, we can generate security measure and safety recommendation for preventing unacceptable losses. Those causal factors are represented by the entity *Causal Factor* in our STAMP-based ontology. *Causal Factor* consists of generic factors that together with *Hazardous Control Action* leads to *Hazard*. In the same sense, we have the entity *Vulnerability*, which is a weakness that can lead to a breach of security in presence of a threat (Rosa et al. [10]).

One of the main outcomes of STPA-Sec is the security measure and safety recommendation that are established by the entity *Recommendation*. We specialize *Recommendation* into two subclasses, name *SafetyRecommendation* and *SecurityMeasure*. *SafetyRecommendation* is the recommendation or mechanism to mitigate the causal factors identified in *SafetyScenario*; while *SecurityMeasure* addresses causal factors identified in *SecurityScenario*. The security ontologies typically present security measures appropriate in an established architecture; Vasilevskaya [13] also present the ontology element "DefenseStrategy" that has instantiations in terms of "AbstractSecurityBuildBlocking", and implemented as "ConcreteSecurityBuildBlocking" elements (e.g., SSH).

In our proposed ontology the entity *SecurityMeasure* is at a higher abstraction level organized in three classes such as Technical, Operational, and Management based on NIST [19]. Appendix A: Entity Security Measure shows the complete table with classes, and families for the STAMP-based Ontology based on NIST [19].

The Entity *Attack* represents an attempt to improperly use an asset with malicious purpose. We follow the same approach as earlier work, Rosa et al. [10], in which the attack is divided in two classes: *PassiveAttack* (e.g., eavesdropping), and *ActiveAttack* (e.g., brute force). As we are in a concept stage and usually there is no much information about the architecture; we adapted the content from Rosa et al. [10] to abstract the different types of passive and active attacks as listed in Appendix B: Entity Attack.

The rightmost entities in Fig. 2 are specific to security and relate to *SecurityScenario*. The entity *SecurityProperty* assumes the values authenticity, authorization, availability, confidentiality, integrity, non-repudiation, and privacy. The entity *LevelOfThreat* is a qualitative evaluation of the possibility of the *SecurityScenario* taking place [17], which comprises the entities *AutomationLevel, AttackerLocation, MissionPhase Attack*. It can have the following values *extremely low, low, moderate*, high and *very high*. The entity *AutomationLevel* identifies the degree to which the attack is automated. The entity *AttackerLocation* refers to where the attack is located. The attack can be launched from inside or outside or both of security perimeter. The entity *MissionPhaseAttack* denotes in which mission phase the attack can be launched such as operation, manufacturing, or maintenance.

The entities *CausalFactor, SecurityProperty*, and *Attack* pave the way for generating security scenarios. The relations between them enable identifying scenarios other than those identified by the STPA-Sec guidewords. We will illustrate the use of those entities in a use case in Section 4.

Finally, we describe the leftmost elements of the ontology in Fig. 2. According to RTCA [17], *Asset* is a representation of logical and physical resources of the aircraft, which contributes to the airworthiness of the aircraft, including functions, systems, items, data, interfaces, processes and information. NIST [18] defines asset as an item of value to achievement of organizational mission/business objectives. Thus, an *Asset* can be a function, security measure, human, or information. In our ontology, function is realized as part of the *Controller* entity (shown under Fig. 3).

Other entities of the ontology are associated to the functional control structure of STPA-Sec analysis. They are controller, sensor, actuator, and controlled process. We create the entity *STPASecElement* (a subclass of *Asset* in Fig. 2) to represent the elements of mission functional control structure; it can assume *ModelledEntity, Sensor, Actuator, Feedback, ControlAction, ProcessModel*, and *ControlAlgorithm* as illustrated in Fig. 3 (left side). *ProcessModel* can be a mental model when a human being is a *Controller*.

As illustrated in the Fig. 3, a *ModelledEntity* can be an instance of a *Controller* or *Controlled Process*. *Control Action, Process Model*, and *Control Algorithm* are part of a *Controller* while *Feedback* is part of a *Controlled Process*. The right side of Fig. 3, Controller Y is a *Controlled Process* of Controller X, and it is also a *Controller* of Controlled Process W. The other entities on the left side of Fig. 3 are independent from each other; it means that an instance of *Sensor* (e.g., wheel speed sensor), cannot be an instance of a *Controller* (and vice-versa). The same is applicable to *Feedback, Actuator, ProcessModel, ControlAction*, and *ControlAlgorithm* entities.

Fig. 4 shows a more detailed hierarchy of the classes in our ontology. At the center is the complete ontology hierarchy. The relationship between classes yields unknown scenarios, which allows a better coverage of scenarios for safety and security analyses. In the absence of our added elements, the STPA-Sec analysis is restricted to six asset types: actuator, sensor, control action, control algorithm, entity (controller, or controlled process) and process model.

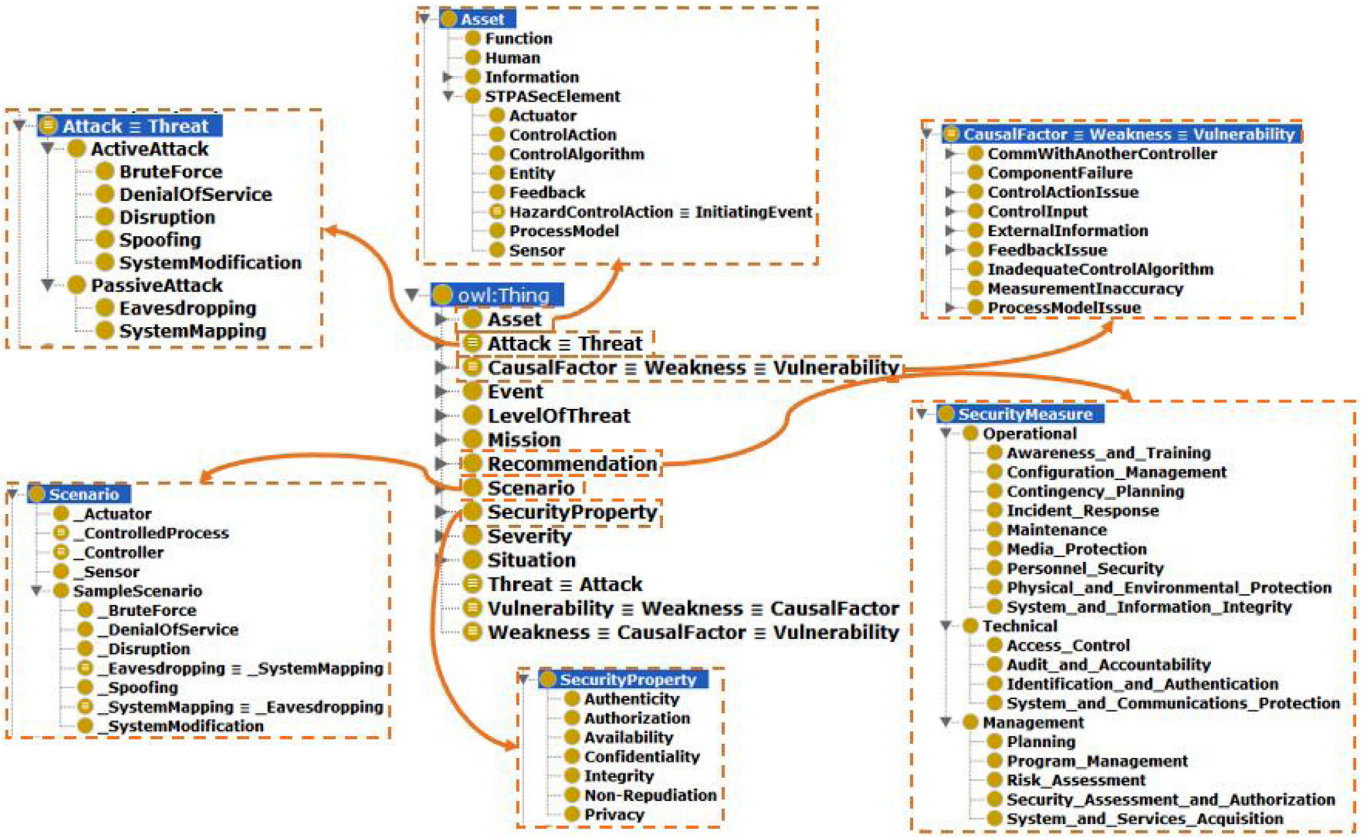

Fig. 4. Detailed STAMP-based ontology hierarchy.

3.2. STAMP-based ontology tool

The knowledge needed to create scenarios is organized by the proposed ontology. Several types of causal factors, attack mechanisms, security properties, and recommendations are recorded as presented in Fig. 4. They can be extended as soon as new types are identified. The reasoning service provides the possible combinations based on the ontology to generate the scenarios according to the attack mechanism or STPA-Sec elements.

We create sample scenarios for each attack mechanism, and then relate them with the causal factors, security properties, and recommendations. This enables us to use the reasoning service to extract distinct possibilities of an attack to damage an asset, which provides the systems engineer with the path to create scenarios. As illustrated in Fig. 5, the class *SampleScenario* is populated manually considering the different attack mechanisms. On the other hand, the reasoning service is in charge of populating the subclasses of *Scenarios* (e.g., *_Controller* class) with the corresponding attacks. For instance, the class *_Controller* is susceptible to attacks such as brute force, denial of service, disruption, eavesdropping, and spoofing.

The UML class model depicted in Fig. 6 represents the structure within SOT to create scenarios. On the left side of Fig. 6, there are six classes and eight relations, which are direct mapping to the elements of the STAMP-based Ontology. The classes *Recommendation, Attack, CausalFactor*, and *SecurityProperty* are already known from Fig. 2. The class *Sample Scenario* records the instances of distinct scenarios. The class *ScenarioCategory* holds the different scenarios categories created in order to assist the systems engineer to discover scenarios in the STPA-Sec analysis. The relation *manages* in the classes *ScenarioCategory* and *CausalFactor* represents that an instance can have an instance for its own class.

The relation between classes allows the systems engineer to select an instance of attack or scenario category, then the tool shows the instances of possible causal factors to be mitigated, possible recommendations to mitigate the causal factors, and the security properties that can be violated. The relation between classes *CausalFactor* and *Attack* is made considering the impact (i.e., causal factors) on the system under analysis when a given attack is launched. Other relation between classes *Attack* and *Recommendation* considers the mechanisms to conduct the attack and how to prevent them.

Table 1 shows the causal factors, recommendations, and security properties suggested when the system under analysis undergoes a *SystemModification* attack. Thereby, we can read as the attack *SystemModification* implies the following causal factors of unacceptable losses *IncorrectProcessModel, InconsistentProcessModel, WrongExternalInformation, WrongInput* and *MeasurementInaccuracy*. The causal factors can be mitigated by recommendations *Identification_and_Authentication, System_and_Information_Integrity, Access-Control, Configuration_Management, Maintenance, Incident_Response* and *System_and_Services_Acquisition*. The *SystemModification* attack threatens security property *Integrity, Confidentiality*, and *Authenticity*.

On the right side of Fig. 6, the model shows four classes to record the scenarios for the STPA-Sec analysis. *CausalScenario* (seen in Fig. 2) requires at least one associated causal factor and one recommendation while *Security Scenario* requires at least one *SecurityProperty*. The class *CausalScenarioAnalysis* contains the attributes scenario description, level of threat, and severity, and it

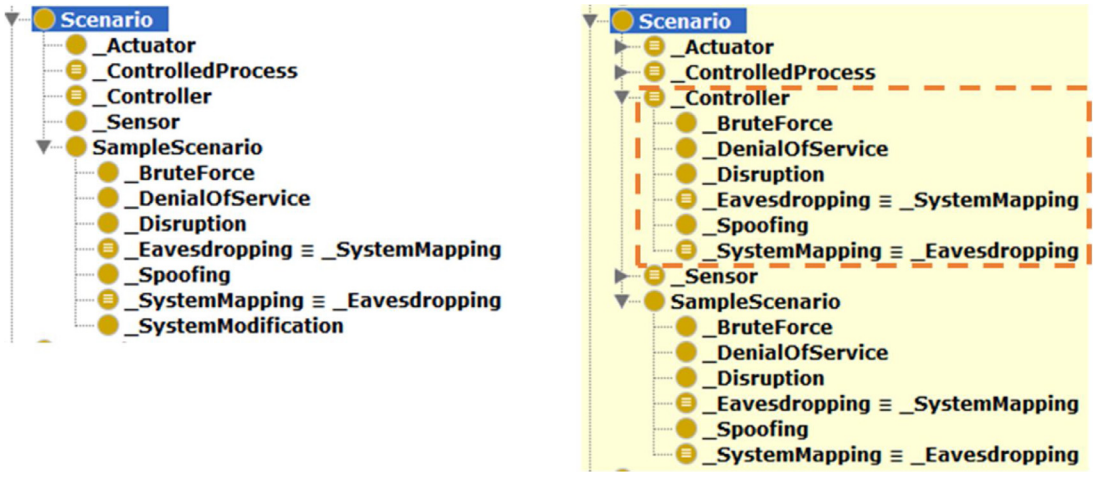

Fig. 5. Reasoning scenarios.

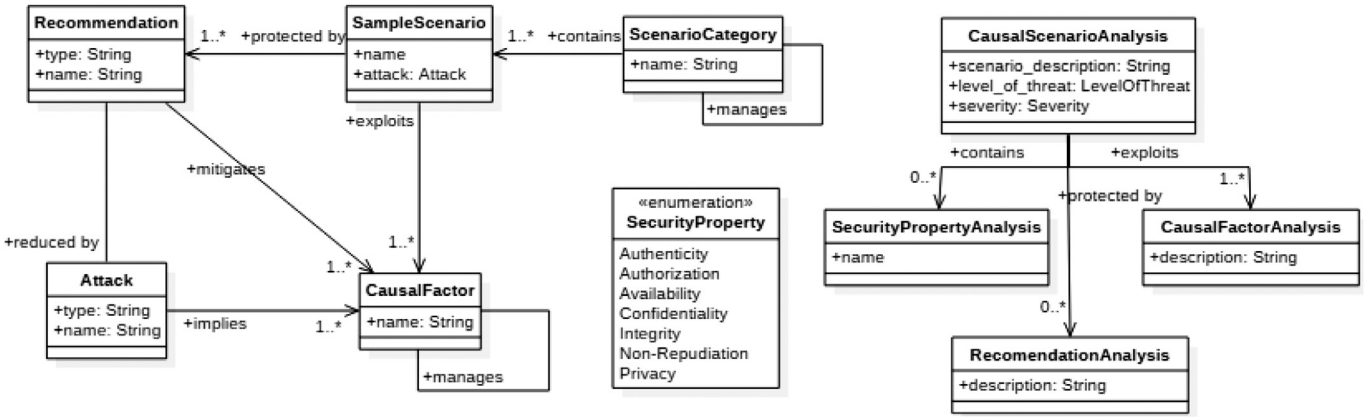

Fig. 6. UML representation of the structure within SOT for scenarios.

Table 1
System modification attack causes.

Causal factor of unacceptable losses	Recommendation	Security property
Incorrect process model	Identification and authentication	Integrity
Inconsistent process model	System and information integrity	Confidentiality
Wrong external information	Access control	Authenticity
Wrong input	Configuration management	
Measurement inaccuracy	Maintenance	
	Incident response	
	System and services acquisition	

relates to the classes *SecurityPropertyAnalysis, CausalFactorAnalysis*, and *RecommendationAnalysis* to record description.

3.3. STAMP-based ontology process

In Fig. 1, the activity *Identify Causal Scenarios* identifies the reasons why hazardous control actions might occur. STPA-Sec tries to guide the systems engineer with guidewords to determine when the system could be at an unexpected state. Guidewords include incorrect feedback, inadequate requirements, design errors, and others.

In safety analysis, we assume that unsafe behavior is due to unintentional actions by benevolent actors leading to inadvertent behavior of the system and its components, considering some scenarios, which can be expressed by guidewords. On the other hand, in security analysis, we have to consider scenarios involving deliberate actions by malevolent actors aiming to cause losses in the system of interest. These scenarios are most complex to describe since they involve attack mechanisms.

Our experience shows that the guidewords in the control loop (right side of Fig. 1) are not enough to assist systems engineer to generate many security scenarios. This is confirmed when we present to systems engineer the likely attack mechanisms that could imply the causal factors, the systems engineer is able to recognize different security scenarios that contribute to an associated HCA. Therefore, we propose process to conduct the analysis using the STAMP-based Ontology presented in Table 2.

SOT provides a systematic way to use the proposed ontology. It has a user interface enabling the systems engineer to conduct the steps depicted in Fig. 2. Fig. 7 shows drop box lists (indicated by "(1)") where the systems engineer selects controller, control action, and the context for the HCA to generate the scenarios. The drop boxes are populated in STPA-Sec Step 1, more specifically in the activity *Identify Hazardous Control Action* in Fig. 1.

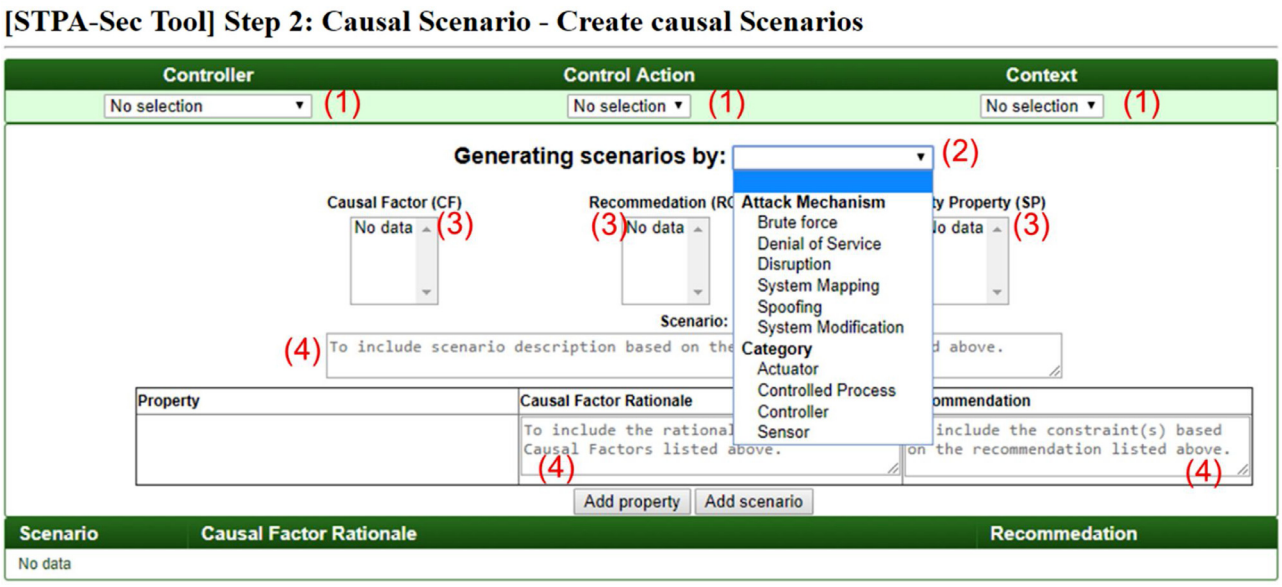

Fig. 7. SOT screenshot.

Table 2
STAMP-based ontology process.

1. Select the HCA to create the scenario.
2. Select the attack mechanism or the category.
   a. The tool shows the possible causal factors, the suggested recommendations, and security properties associated to the chosen attack mechanism.
   b. The systems engineer chooses several options presented by the tool.
3. Select the causal factor, the recommendation, and security property.
   a. In this moment, the tool shows the chosen causal factors, recommendations, and security properties according to the type of system under analysis.
4. Write the scenario, Causal factor rationale, and Recommendation.

The contents of the scenarios drop box (labeled by "(2)") are populated using the STAMP-based Ontology. It contains a list of attack mechanisms and categories that are used to populate the combo box lists labeled by "(3)" in Fig. 7. The systems engineer is able to choose the causal factor, recommendation, and security property to create a scenario. The next section describes a case study using the proposed ontology, tool and process.

4. Application of the proposed approach to an aeronautic case study

This chapter describes the application of the proposed ontology, tool, and process in a case study.

4.1. System of interest – aircraft system

We choose an aircraft system that enables the avionic systems to update its database and software through wireless connection. This feature reduces cost and time, but potentially introduces cybersecurity vulnerabilities that affect aircraft safety.

The avionic system contains navigation databases. For the system of interest, the navigation database contains elements from which the flight plan is constructed. The navigation database capacity has always been an issue in the aircraft Flight Management System (FMS). FMS provides the primary navigation, flight planning, and optimized terminal routes and en route guidance for the aircraft. It is typically comprised of interrelated functions such as navigation, flight planning, trajectory prediction, performance computations, and guidance.

FMS and associated databases are an essential part of modern airline avionics. The navigation database is normally updated every 28 days. Thus, the database update is a constant task, and the maintenance usually has to download the database from database suppliers, and then update the aircraft system. Today, the maintenance operation takes hours to download due to the database size. It is predicted that the database size will increase approximately −8% annually for the near future [6]. At first impression the new functionality would not affect the installed system in certified aircrafts.

Fig. 8 shows our system of interest. An equipment that is added into the avionic system provides a wireless interface to communicate with a portable electronic device (PED) which allows transferring of database contents and software. From safety perspective, it seems that there is no impact, however this connection open doors to several security attacks. The first raised concern is the addition of a device from an Airline Information Services Domain (AISD), on the right side of Fig. 8, to provide data to the Aircraft Control Domain (ACD), on the left side of Fig. 8. Therefore, with the PED, a security threat can potentially impact the aircraft control. In the next section, we analyze those interactions focusing on the security scenarios.

4.2. STPA-Sec analysis through proposed approach

The application of the first activities of STPA-Sec are briefly described in this section. We conducted the Step 1 of STPA-Sec according to the original guidelines in order to use the outcome to perform Step 2, which is the focus of this work. The activity *Identify Systems Engineering Foundation* establishes a context for the safety and security analyses. The context includes identifying system purpose and scope, identifying assumptions and constraints associated with the analysis, identifying unacceptable losses, hazards and system boundaries, and model the mission functional control structure (MFCS). The system of interest is a:

"Wireless communication system to provide *operational flight information* by means of *sharing and updating operational flight information* in order to contribute to *flight and maintenance of the aircraft*."

Fig. 8. System of interest.

Table 3
Unacceptable losses and hazards.

	L1: Loss of navigation information to crew and passenger	L2: Loss of Maintenance support for flight operation data (e.g., FP, software, and database)	L3: Loss of protection against unauthorized electronic interaction
H1: Unable to display navigation information	X		X
H2: Uncertainty about flight operation data		X	X
H3: Disclosure of flight operation data		X	X

The identification of unacceptable losses and hazards comes from the customers and other stakeholders. From the safety perspective, an unacceptable loss results in some loss that is unacceptable to the stakeholders such as death or injury. From the security perspective, the systems engineer tries to identify the essential system services and functions to be protected and controlled. The list of unacceptable losses may have losses that stem from both safety and security concerns, from safety concern only or from security concern only. The unacceptable losses are associated to hazards. Sometimes such associations are not straightforward to conceive. Table 3 shows the unacceptable losses, the hazards, and the relations between them; for instance, H1 (Unable to display navigation information) can lead to losses L1 (Loss of navigation information to crew and passenger) and L3 (Loss of protection against unauthorized electronic interaction). L1 and L2 can be seen as loss of aircraft and human lives while L3 can be seen as loss of reputation of airline and regulatory agencies, and loss of aircraft and loss of human lives, depending on the extension of the attack.

The last task in the activity *Identify Systems Engineering Foundation* is to model the MFCS. The MFCS provides a graphical representation of the functional concept of the system being analyzed where the system boundaries are delimited. It is a hierarchical structure of elements where each element imposes constraints on the activity of lower level elements. Fig. 9 illustrates the MFCS of our system of interest. The control actions (down arrow) and feedbacks (up arrow) are shown.

The next activity is *Identify* hazardous control actions in the mission functional control structure. It consists of identifying whether control actions given incorrectly, out of sequence/later/early, or missing may lead the system to a hazardous state. Each control action and the ways of provision of control action must be analyzed. Among many control actions possibilities that we considered, in

Table 4 we list two control actions from Pilot to PED and one control action from Pilot to avionic system where we identified the HCAs. Both *not providing causes hazard* and *providing causes hazard* are illustrated. The description of providing the control action (columns) *too early, too late, out of order, stopped too soon*, and *applied too long* are not shown here because we considered theses as not critical for these control actions. In addition, we select the control action *download the database* and its hazardous control actions to use as running example in the identification of causal scenarios.

The third and last activity is *Identify Causal Scenarios* where our proposed ontology approach is applied. We begin by considering controllers, controlled processes and control actions (from

Table 4
Excerpt of HCAs.

Control Action	Not providing causes hazard	Providing causes hazard
Send Flight Plan	[HCA-1] when avionics standby flight plan is missing or outdated (H1)	[HCA-2] when PED flight plan is corrupted (H1, H2) [HCA-3] when PED is not authorized to connect (H2) [HCA-4] when PED flight plan is tampered (H2)
Download database	[HCA-5] when database in the avionic systems is out-of-date - (H1)	[HCA-6] when database is wrong (H1) [HCA-7] when database is corrupted (H1) [HCA-8] when database source is not trusted (H2) [HCA-9] when database is tampered – (H1)
Update database	[HCA-12] when standby database is missing or outdated (H2)	[HCA-13] when PED is not allowed (H2) [HCA-14] when standby database is corrupted (H2) [HCA-15] when aircraft is in flight (H2) [HCA-16] when standby database is tampered (H2)

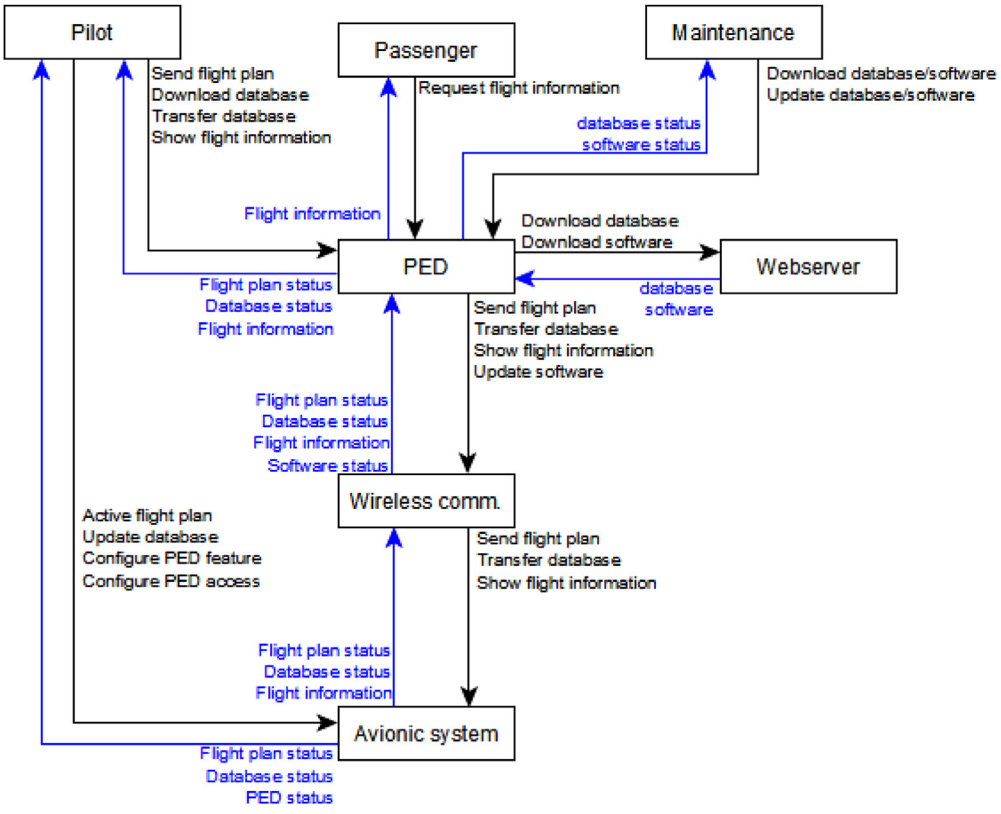

Fig. 9. Mission functional control structure.

mission functional control structure), and the identified HCAs from STPA-Sec Step 1. All the information must be stored in the instantiation of the ontology to be used by the tool. The scenario we generated is shown in Fig. 10. The retrieved data is captured from the previous STPA-Sec step (i.e. *Identify systems engineering foundation* and *Identify* hazardous control actions).

The generation of scenarios is made in two ways, by attack mechanisms or categories. Selecting the Controller with category, a list of causal factors are presented. The first scenario that comes up is that the pilot can select a wrong data from database due to incomplete process model. When the causal factors relate to the attack mechanisms we have a security scenario.

From the security perspective, a typical systems engineer has difficulty to generate security scenarios. When the attack mechanisms are presented, for instance *SystemModification*, other causal factors are presented by the tool. Thus, the attack *SystemModification* can imply in an incorrect process model. Table 5 shows the causal and security scenarios for the HCA-6 (when data from database is wrong). Causal scenario contains the causal factor (CF) property. Security Scenario contains the causal factor (CF), recommendation (RC), and security property (SP).

Appendix C: Screenshot shows the screenshots of SOT to create the causal and security scenarios presented in Table 5. In Appendix C.1, the causal scenario is created; the causal factor *IncompleteProcessModel* is selected, and the three fields are filled. The second scenario is presented in Appendix C.2, the attack mechanism *SystemModification* is selected; and then the causal factor *WrongExternalInformation*, recommendation *Identification_and_Authentication*, and security property *Authenticity*. Appendix C.3 shows the list of two scenarios recorded.

5. Evaluation of the proposed approach

In this section, we describe an initial evaluation of the proposed approach as a proof of concept. We compare the results of analyses made by two systems engineers. Both engineers received the results of the analyses in the initial STPA activities: *Identify Systems Engineering foundations* and *Identify* hazardous control actions (Step 1), and then performed the third activity *Identify Causal Scenarios*. We analyzed the completeness of the results obtained by the systems engineers against each other. The first engineer conducted the analysis without the tool while the second engineer employed the tool to support this process.

The two systems engineers had similar expertise, they work in the same engineering group and are familiar with how the system of interest works. For the systems engineer that used the SOT,

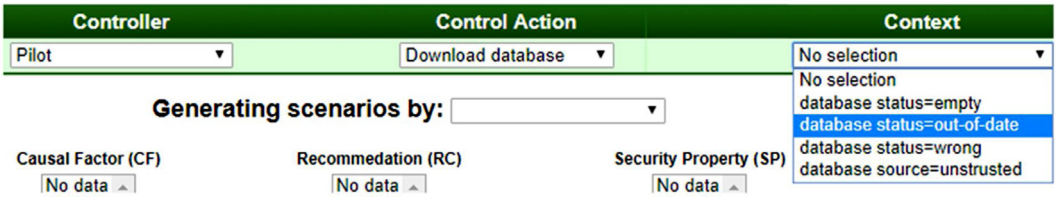

Fig. 10. SOT - select HCA.

Table 5
Causal and security scenario for HCA-6.

[Causal scenario] TS-6.1: Pilot is not aware what type of database must be downloaded when he is connected to the database server. Different databases are listed [Inconsistent Process Model].		
Property	*Causal factor rationale*	*Recommendation*
CF: Incomplete process model	*[Incomplete process model]* Process model does not identify database and aircraft types.	Pilot shall be trained to identify the type of database according to the aircraft and system model.
[Security scenario] TS-6.2: Pilot is connected to the database server and requests to download the database. The communication is spoofed by an adversary that sends another file to the Pilot PED [Wrong communication, Active=>Spoofing, Authenticity].		
Property	*Causal factor rationale*	*Recommendation*
CF: Wrong communication RC: Identification and authentication SP: Authenticity	*[Wrong communication]* No authentication between PED and Webserver.	*[Identification and authentication]* PED shall establish a security channel with webserver.
[Security Scenario] TS-6.3: An adversary modifies the installed application in the PED, and a wrong database is made available to the pilot. [Wrong Input, Active=>System modification, Integrity].		
CF: Wrong input RC: System and information integrity SP: Integrity	*[Wrong Input]* Missing integrity check.	*[System and information integrity]* The PED shall verify integrity of configuration file when the configuration is loaded to memory.

Table 6
Threat scenarios using or not the proposed tool.

Systems engineer	Number of HCAs considered	Number of threat Scenarios Identified	Number of HCAs not addressed
Who used only STPA-Sec	60	47	16
Who used STPA-Sec with the Ontology tool	60	78	0

we explained how to use it considering the attack mechanisms. The outcome was analyzed by quantifying the number of threat scenarios identified by each systems engineer.

The summary of outcomes of analyses by the engineer who used only STPA-Sec and the one that used STPA-Sec with the ontology tool are shown in Table 6. From the 60 HCAs, 47 threat scenarios were identified by systems engineer using basic STPA-Sec while 78 threat scenarios were identified by systems engineer using SOT. The systems engineer who used the basic STPA-Sec was not able to address 16 hazardous control actions. There are several attacks, including denial of service attack, system modification (tampered data) attack, eavesdropping attack that were not considered by the systems engineer following the basic STPA-Sec.

Table 7 lists the threat scenarios (TS) that shows the results of the evaluation. For some HCAs more than one TS was generated. The TS highlighted in blue/italic are the ones created by the systems engineer using SOT, while the others were identified by both systems engineers. The evaluation revealed that 9 new TS corresponding to a denial of service attack that are not typically considered in a safety analysis, and when the proposed tool shows this type of attack mechanism as an option, the systems engineer is encouraged to think about such an attack. In the same way, the passive attack eavesdropping was identified five times.

The system modification attack was exploited by the systems engineer without SOT. However, some threat scenarios were not captured. In discussing with the systems engineer, the engineer mentioned lack of familiarity with that type of attack, and when the tool showed the relation between causal factors and attack mechanism system modification, the systems engineer were able to identify 12 new threat scenarios in addition to the other 12 generated previously. Following the same principle, six TSs, covering spoofing attacks, were also identified not using and five TS using SOT.

Regarding safety scenarios, the approach did not show any advantage. However, we consider that this was according to expectations. First, the ontology was not populated with causal factors that range over (benign) failures, and second, both engineers were experienced in analyzing classic safety-related causes so our focus was enhancing their understanding of impacts of security on safety. The ontology approach is clearly exhibiting the potential to support the systems engineer during generation of threat scenarios.

6. Concluding remarks and future works

In this section we summarize, discuss and conclude the paper with some directions for future works. We proposed an ontology-based approach to support STPA-Sec analysis. The ontology-based approach is concretized by a tool that aids the systems engineer to identify safety and security scenarios and recommendations. The tool enacts a process to describe how to feed and retrieve information from a knowledge repository. We employed the proposed approach to analyze an aeronautic system and we showed that the approach helps the systems engineer to identify more security scenarios. We also showed that some hazardous control actions are not addressed if the systems engineer uses only the basic STPA-Sec, without the ontology tool.

6.1. Discussion

When creating the ontology, we realized that distinct ontologies sometimes use the same concept for different entities. Zhou et al. [15,16] brings the definition of *InitiatingEvent* as an undesirable or unexpected event that can lead to a Hazard. *InitiatingEvent* and *Hazard* are equivalents and used interchangeably by our work. The entities *CausalFactor* and *Vulnerability* are considered as equivalent and used interchangeably in our ontology. In addition, Rosa et al. [9] bring Weakness as another equivalent entity to Vulnerability.

Another entity in our ontology that reuses the same concept from other ontologies is *SecurityMeasure*. In other ontologies (Zhou et al. [15,16], and Rosa et al. [9]) we find the terms *Countermeasure*, and *SecurityRecommendation*. The latter shares the same meaning in our ontology and the former is equivalent to our term *SecurityMeasure*.

In addition, the approach proposed by Goluch et al. [4] is similar to our approach regardless the application domain. The STAMP-based ontology approach considers safety and security assessment in the concept stage, while Goluch et al. [4] approach covers physical infrastructure elements. Besides, our approach allows to perform top-down analysis, instead of bottom-up.

Table 7
Evaluation of threat scenario.

HCA	Threat Scenarios
HCA-1	TS-1.1: Pilot is not aware that when the standby flight plan is missing or outdated, it is required to update or send a new flight plan [Incorrect process model]. TS-1.2: Pilot is not aware the PED application is modified, and the correct message is not displayed in the PED to show that the standby flight plan is missing or outdated, and no action is performed for that [Wrong External Information, Active=>System modification, Integrity]. TS-1.3*: Pilot is not aware the PED application is connected in a wrong server device, and he/ she sends to Wireless Communication the flight plan [Unauthorized communication, Passive=>System mapping, Authorization].*
HCA-2	TS-2.1: Pilot sends the flight plan before verifying that the flight plan is correct [Inadequate control algorithm]. TS-2.2: Pilot is not aware that the recorded flight plan is modified by a malicious software, and he sends a modified flight plan [Inconsistent process model, Active=>System modification, Integrity].
HCA-3	TS-3.1: Pilot uses an unauthorized PED to connect into the avionic systems and sends the flight plan [Incorrect process model]. TS-3.2: An adversary intercepts the communication between authorized PED and avionics system. Then, the adversary sends a modified flight plan [Malformed operation, Active=>Spoofing, Authenticity].
HCA-4	TS-4.1: Pilot is not aware the recorded flight plan is modified by an adversary, and sends a modified flight plan [Inconsistent process model, Active=>System modification, Integrity].
HCA-5	TS-5.1: Pilot is not aware the flight plan needs to be updated. PED does show this information [Missing feedback]. TS-5.2*: An adversary attacks the PED preventing the authorized PED to communicate to the avionics system to receive the warning that the database is missing or outdated. Pilot is unable to send the command [Missing communication, Active=>Denial of service, Availability].*
HCA-6	TS-6.1: Pilot is not aware what type of database must be downloaded when he is connected to the database server. Different databases are listed [Inconsistent Process Model]. TS-6.2: Pilot is connected to the database server and requests to download the database. The communication is spoofed by an adversary that sends another file to the Pilot PED [Wrong communication, Active=>Spoofing, Authenticity]. TS-6.3*: An adversary modifies the installed application in the PED, and a wrong database is made available to the pilot. [Wrong Input, Active=>System modification, Integrity]*
HCA-7	TS-7.1: Pilot starts the command to download the database, however the download is interrupted before it is completed [Wrong Input, Active=>System modification, Integrity]. TS-7.2*: Pilot starts the command to download the database, however the PED is infected by a malware that modifies the downloaded files [Wrong Input, Active=>System modification, Integrity].*
HCA-8	TS-8.1: An adversary modifies the PED system so that the pilot accesses a wrong database server. Thus, the pilot is not aware that the accessed webserver is not trusted, and download a malicious database [Wrong External Information, Active=>Spoofing, Authenticity].
HCA-9	TS-9.1*: An adversary tampers a database; the tampered database is inserted during a spoofing attack. Pilot is unaware that the database is tampered, and downloads the file [Wrong Input, Active=>System modification, Integrity].*
HCA-10	TS-10.1: Pilot is unaware he/she needs to enable system to show flight information [Incomplete process model].
HCA-11	TS-11.1*: An adversary is connected to the avionics system. Pilot authorizes to show flight information, but he/she is unaware that an adversary is receiving flight information [Inadequate control algorithm, Passive=>System mapping, Confidentiality].*
HCA-12	TS-12.1: Pilot is not aware that when the standby database is missing or outdated, it is required to update and no command is issued [Incorrect process model]. TS-12.2*: Pilot is not aware that the PED application is modified, and the correct message is not displayed in the PED to show that the standby database is missing or outdated, and no action is performed for that [Wrong external information, Active=>System modification, Integrity].*
HCA-13	TS-13.1*: An adversary is connected to the avionics system with a tampered database to be downloaded. When the Pilot confirms to update the database, an unexpected database is updated [Unauthorized communication, Active=>Spoofing, Authenticity].*
HCA-14	TS-14.1: Pilot updates the database before verifying that the database is correct [Inadequate control algorithm]. TS-14.2*: Pilot is not aware the recorded database is modified by a malicious software, and requests to update the database [Inconsistent process model, Active=>System modification, Integrity]*
HCA-15	TS-15.1: Pilot is not aware that the database must be updated in flight, and connects to the PED with a valid database and then requests to update [Inappropriate control action].
HCA-16	TS-16.1*: Pilot checks the available databases. The database stored is tampered. The Pilot issues the command to update [Wrong Input, Active=>System modification, Integrity].*
HCA-17	TS-17.1: Pilot is not aware that he/she needs to authorize PED to communicate with the Wireless Communication and no authorization is granted [Incorrect process model].
HCA-18	TS-18.1: Pilot is not aware the procedure to grant PED authorization, and he allows access to all PED accesses the Wireless Communication [Incomplete Process Model]. TS-18.2: An adversary gets access to the Wireless Communication and modifies the configuration file that contains the authorized devices. New devices get access and other accesses are removed [Inappropriate control action, Active=>Disruption, Availability].
HCA-19	TS-19.1*: Pilot grants authorization to a PED but, Pilot is unaware that the PED is from an adversary that is stealing data [Unauthorized communication, Passive=>Eavesdropping, Privacy]*
HCA-20	TS-20.1: Maintenance personnel is unaware that there is a new version of database available. PED does not show this information [Missing feedback] TS-20.2*: An adversary attacks the PED preventing the authorized PED to communicate with the webserver and no warning message is presented. Maintenance is unable to download the firmware [Missing communication, Active=>Denial of service, Availability]*
HCA-21	TS-21.1: Maintenance personnel is not aware what type of firmware must be downloaded when he/she is connected to the web server. Different firmware are listed [Inconsistent Process Model] TS-21.2*: Maintenance is connected to the webserver and requests to download the firmware The communication is spoofed by an adversary that sends another file to the maintenance PED [Wrong communication, Active=>Spoofing, Authenticity].*
HCA-22	TS-22.1: Maintenance personnel starts the command to download the database, however the download is interrupted before it is completed [Wrong Input, Active=>System modification, Integrity]
HCA-23	TS-23.1: Maintenance personnel is not aware the firmware is modified by an adversary, and downloads the file [Wrong Input, Active=>System modification, Integrity].
HCA-24	TS-24.1: An adversary modifies the PED system to access a wrong webserver. Thus, the Maintenance personnel is not aware that the accessed webserver is not trusted, and downloads a malicious tampered firmware [Wrong External Information, Active=>Spoofing, Authenticity].
HCA-25	TS-25.1: An adversary installs a malware in the maintenance personnel's PED that prevents the PED to send the command to update the firmware. Maintenance personnel is unable to update the firmware [Delayed operation, Active=>Denial of service, Availability]. TS-25.2*: An adversary provides intentional interference attacks on wireless networks. The attack prevents the Maintenance to update the firmware [Missing Communication, Active=>Denial of service, Availability].*

(*continued on next page*)

Table 7 (*continued*)

HCA	Threat Scenarios
HCA-26	TS-26.1: Maintenance personnel needs to dispatch the airplane quickly, and therefore maintenance personnel starts to update the database. While the database is updated, the maintenance personnel also updates the firmware [Incomplete Process Model]. TS-26.2: Maintenance personnel issues a command to update the firmware. An adversary at the same moment gets access to the wireless communication and sends a modified database [Inadequate Control Algorithm, Active=>Disruption, Availability].
HCA-27	TS-27.1*: An adversary, during flight, gets access to the wireless communication. He/she has a modified firmware in his/her computer, and then issues a command to update the firmware in flight [Inadequate Control Algorithm, Active=>Spoofing, Authenticity].*
HCA-28	TS-28.1: Maintenance personnel does not verify the PED battery and starts to update the firmware [Inadequate Control Algorithm].
HCA-29	TS-29.1: An adversary invades (attacks) the maintenance PED and replaces the firmware source with a tampered firmware. The maintenance personnel is unaware and updates using tampered firmware [Wrong External Information, Active=>System modification, Integrity].
HCA-30	TS-30.1*: Maintenance personnel connects to the Wireless Communication and sends the command to update the firmware. An adversary starts doing several requests to connect to the Wireless Communication until the system goes down [Missing communication, Active=>Denial of service, Availability].*
HCA-31	TS-31.1: Maintenance personnel has downloaded several files. When the maintenance personnel connects to the Wireless Communication to update, he/she is not sure which file to use, and chooses the wrong one [Wrong Input].
HCA-32	TS-32*: Maintenance personnel has downloaded the firmware. The download is completed. An adversary has installed a malware in his/her PED which replaces the downloaded firmware with a tampered firmware. The maintenance personnel is unaware and issues the command to update the firmware [Wrong Input, Active=>System modification, Integrity].*
HCA-33	TS-33: Maintenance personnel has downloaded the firmware. The download is complete. An adversary has installed a malware that modified all files in the PED. The maintenance personnel is unaware and issues the command to update the firmware [Wrong Input, Active=>System modification, Integrity].
HCA-34	TS-34.1*: An adversary performs intentional interference attacks on wireless networks. The attack prevents the PED to connect to the Webserver [Missing Communication, Active=>Denial of service, Availability].* TS-34.2*: An adversary has installed a malware that prevents the PED to connect to the webserver. PED does not identify any available connection to communicate to the webserver [Missing Communication, Active=>Denial of service, Availability].*
HCA-35	TS-35.1*: An adversary modifies the initial configuration of the PED. The list of proper database is altered. Then, the PED downloads the wrong database [Wrong Input, Active=>System modification, Integrity].*
HCA-36	TS-36.1: Pilot starts the command to download the database, however the download is interrupted before it is completed [Wrong Input, Active=>System modification, Integrity].
HCA-37	TS-37.1: Pilot is not aware that the database is modified by an adversary, and downloads the file to his/her PED [Wrong Input].
HCA-38	TS-38.1*: An adversary invades the Pilot's PED and modifies the configuration to access a fake webserver. When PED is requested to download the database, a tampered database is provided [Wrong External Information, Active=>System modification, Integrity].*
HCA-39	TS-39.1*: An adversary performs intentional interference attacks on wireless networks. The attack prevents the PED to connect to the Webserver [Missing Communication, Active=>Denial of service, Availability].* TS-39.2*: An adversary has installed a malware that prevents the PED to connect to the webserver. PED does not identify any available connection to communicate to the webserver [Missing Communication, Active=>Denial of service, Availability].*
HCA-40	TS-40.1: PED is unaware that the battery is low, and starts downloading the firmware. No warning is presented to inform that the charge is not enough to complete the download [Incomplete Process Model].
HCA-41	TS-41.1*: An adversary invades (attacks) the Pilot's PED and modifies the configuration to access a fake webserver. When PED is requested to download the database, a tampered firmware is provided [Wrong External Information, Active=>System modification, Integrity].*
HCA-42	TS-42.1: The PED algorithm does not verify if a connection is established before sending the flight plan, and no warning is provided [Inadequate Control Algorithm]. TS-42.2: An adversary modifies the initial configuration of the PED. Then, the PED is unable to connect to the Wireless Communication [Wrong Communication, Active=>System modification, Integrity].
HCA-43	TS-43.1: The PED algorithm does not receive any feedback information that there is enough bandwidth in the Wireless Communication, and tries to send the flight plan when there is no space [Inadequate Control Algorithm].
HCA-44	TS-44.1: An adversary connects to the Wireless Communication as an authorized PED. A fake flight plan is prepared and sent to the Wireless Communication [Unauthorized communication, Active=>Spoofing, Authenticity].
HCA-45	TS-45.1*: An adversary installs a malware in the Pilot's PED that modifies the Flight Plan when stored. The PED seems to send the correct flight plan, but the flight plan is modified first [Wrong External Information, Active=>System modification, Integrity].*
HCA-46	TS-46.1*: Pilot connects to the Wireless Communication and sends the command to transfer the Flight Plan. An adversary starts doing several requests to connect to the Wireless Communication until the system goes down [Missing communication, Active=>Denial of service, Availability].*
HCA-47	TS-47.1: When PED receives the command to send a flight plan, the PED does not verify the file integrity, and sends a corrupted flight plan [Inadequate Control Algorithm].
HCA-48	TS-48.1: The PED algorithm does not verify PED memory space, and sends the command to receive the flight plan. The transfer does not complete due to lack of space in the PED [Inadequate Control Algorithm].
HCA-49	TS-49.1*: An adversary connects to the Wireless Communication as an authorized PED. Once connected, he/she starts receiving several flight plans from Wireless Communication [Unauthorized Communication, Passive=>Eavesdropping, Confidentiality].*
HCA-50	TS-50.1: The PED algorithm does not verify if a connection is established before sending the database, and no warning is provided [Inadequate Control Algorithm].
HCA-51	TS-51.1: The PED algorithm does not receive any feedback that there is no free space from Wireless Communication, and tries to send the database when there is no space [Inadequate Control Algorithm].
HCA-52	TS-52.1: When PED receives the command to send a database, the PED does not verify the file integrity, and then a corrupted database is sent [Inadequate Control Algorithm].
HCA-53	TS-53.1: An adversary connects to the Wireless Communication as an authorized PED. A fake database is prepared and sent to the Wireless Communication [Unauthorized communication, Active=>Spoofing, Authenticity].
HCA-54	TS-54.1: An adversary installs a malware in the Pilot's PED that modifies the database when stored. The PED seems to send the correct database, but the database is modified first [Wrong External Information, Active=>System modification, Integrity].
HCA-55	TS-55.1: The PED algorithm does not verify if a connection is established before updating the Wireless Communication firmware, and no warning is provided [Inadequate Control Algorithm]. TS-55.2: *An adversary modifies the initial configuration of the PED. Then, the PED is unable to connect to the Wireless Communication in order to update the Wireless Communication firmware [Wrong Communication, Active=>System modification, Integrity].*
HCA-56	TS-56.1: An adversary installs a malware in the Pilot's PED that modifies the Wireless Communication firmware when stored. The PED seems to send the correct Wireless Communication firmware, but the Wireless Communication firmware is modified first [Wrong External Information, Active=>System modification, Integrity]. TS-56.2*: An adversary connects to the Wireless Communication as an authorized PED. A tampered Wireless Communication firmware is prepared and sent for updating [Unauthorized Communication, Active=>Spoofing, Authenticity].*

(*continued on next page*)

Table 7 (*continued*)

HCA	Threat Scenarios
HCA-57	TS-57.1: *An adversary, during flight, gets access to the Wireless Communication. He/she has a modified firmware in the computer, and then issues a command to update the firmware in flight [Inadequate Control Algorithm, Active=>Spoofing, Authenticity].*
HCA-58	TS-58.1: PED does not have any information that the database/flight plan has been transferred by another PED. Then, PED starts to update the Wireless Communication firmware [Incomplete Process Model].
HCA-59	TS-59.1: When PED receives the command to update the Wireless Communication firmware, the PED does not verify the file integrity, and then a corrupted firmware is sent [Inadequate Control Algorithm].
HCA-60	TS-60.1: *An adversary connects to the Wireless Communication as an authorized PED and the Wireless Communication starts sending supplementary information to an unauthorized device [Unauthorized Communication, Passive=>Eavesdropping, Confidentiality].*

The proposed STAMP-based ontology considers safety and security concerns; a holistic analysis is conducted to consider the system of interest, and its environment. Our approach gives an opportunity to analyze security and safety together to identify the risks. For security, our approach based on ontology allows the systems engineer to consider each HCA against relevant and distinct attack mechanisms.

The STPA-Sec analysis approach identifies controllers (entities), control actions, and contexts that lead to hazards, which are found in other ontologies [15]. However, our ontology contains additional entities such as mission (and relates it to the method, purpose, and goal), scenario (circumstances leading to hazards), causal factor (factor necessary within a scenario), safety and security recommendation (means to handle the causal factors), and other entities to cover security concerns.

STPA-Sec inherits the control loop construct from STPA that assists the engineer in generating safety-related scenarios. However, the control loop is very abstract and leaves out details that can be useful in security-related scenario generation. In addition, concepts for identifying security-specific concerns are missing there. In our case study, we noted that our approach enhances the capability of the systems engineer to identify threat scenarios not captured by the basic STPA-Sec. This can be exemplified, for instance when the controller *Wireless Communication* requests several data types from the controlled process *avionic system* (see TS-46.1). SOT also supports the systems engineer when identifying different types of security measures.

The proposed approach based on ontology provides all the possible scenarios given the stored knowledge on attack mechanisms, assumptions made, and the elaborated models. The systems engineer must analyze all the scenarios in order to both identify similar scenarios and identify a reduced list of scenarios that achieves an acceptable level of coverage.

There is an initiation effort to use SOT for the first time, which is related to the preparation of the relation between causal factors and attack mechanisms. The user has to extract the relation between them from the ontology and then create the records into the database from which SOT retrieves the information. The relation changes when causal factors or attack mechanisms are added to the ontology.

In addition, the user has to enter manually the information obtained in the first two activities of STPA-Sec such as controller, controlled process, control actions and HCAs, in order to start the STPA-Sec Step 2 with the ontology tool. We intend to integrate the ontology tool in WebSTAMP. WebSTAMP will then provide the functionalities to conduct both activities of STPA-Sec.

In our experience, a critical task of the proposed approach performed by the systems engineer is to create the relationship between causal factors and attack mechanisms. Cybersecurity experts are required to determine which attack mechanisms can imply a causal factor. This is not an easy task because they should have a broader view of the system than one that only considers security. Based on this knowledge, SOT is able to support deriving the threat scenarios and recommendations.

6.2. Conclusions and future works

The STAMP-based ontology for safety and security is to our knowledge the first attempt to join the two disciplines; it also includes the STAMP elements not present in other ontologies. The relationships among causal factors, recommendations, and security properties allow the systems engineer to identify distinct scenarios. The ontology can also be modularly extended to include new information (e.g., new attack types, new security mechanisms). Our preliminary evaluation shows that the STAMP-based ontology tool SOT and the process fill the gap for generating threat scenarios and recommendations in a systematic way, and identifying the attack mechanisms and security measures.

Any method for safety and security analysis is limited to the information encoded in terms of assumptions and knowledge provided in the models or ontologies. Our approach is not different. What our tool does is providing a systematic means of going through the available knowledge and finding hazardous scenarios that could have been overlooked, as well as hinting known mitigations that may help the systems engineer in evaluation of security cost effectiveness.

SOT is intended to become part of a framework, named WebSTAMP, a web application for STPA & STPA-Sec (Souza et al. [12]). This feature should be implemented soon into WebSTAMP. With flexibility in mind, our current works employs the Laravel Framework for web artisans. Laravel Framework supports Model-View-Controller that keeps clarification between logic and the presentation, integrates easily with third-party libraries (e.g., composer), and manages the database structure.

There are some future works that are of interest. After creating security scenarios, the categorization of the level of threat is one of the most difficult tasks during the analysis. The categorization allows prioritizing the highest severe threat scenarios. We are working on a proposal to support the systems engineer to categorize the security scenarios.

Another possible investigation is assess the ability to reuse the analysis using the ontology-based approach. For instance, we conducted an analysis of the aircraft level with PED, identifying the hazardous control actions, threat scenarios, and recommendations. If we have to perform a new analysis of the same system in a different aircraft but with additional functionalities, we hypothesize that we can reuse the knowledge generated by current work. The reusability issue is interesting topic to further study.

Conflict of interest

On the cover sheet of this submission, all authors must describe any financial or personal relationship that could cause a conflict of interest regarding this article. Examples of financial or personal relationships with people or organizations that could inappropriately bias your work include employment, consultancies, stock ownership, honoraria, paid expert testimony, patent applications/registrations, and grants or other funding.

The authors declare no conflict of interest.

Acknowledgements

The work of the second author was supported by CNPq (grant numbers 403921/2016-3 and 306186/2018-7) and CISB SAAB (call 04/2016). The work of the last author was supported by the national projects on aeronautics (NFFP7-04890) and the research centre on Resilient Information and Control Systems (www.rics.se).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.jisa.2019.05.014.

Appendix A. Entity Security Measure

Table A.1
Security Measures for the STAMP-based Ontology based on NIST [19].

Class	Family	Description
Technical	Access control	Access control family comprises access control policy and procedures, account management system use notification, remote access, wireless access, access control for mobile devices, use of external information systems, and so on.
Technical	Audit and accountability	Audit and accountability comprises audit and accountability policy and procedures, auditable events audit review, audit analysis, audit reporting, monitoring for information disclosure, and so on.
Operational	Awareness and training	Awareness and training comprises security awareness and training policy and procedures, security training, security education and training for organizational personnel, and update with latest recommended security practices, techniques, and technologies.
Operational	Configuration management	Configuration management comprises configuration management policy and procedures, configuration change control, security impact analysis, and information system component inventory.
Operational	Contingency planning	Contingency planning comprises contingency planning policy and procedures, contingency training, alternate storage site, information system backup, information system recovery and reconstitution.
Technical	Identification and authentication	Identification and authentication family comprises identification and authentication policy and procedures, device identification and authentication, identifier management, authenticator management and feedback, and cryptographic module authentication.
Operational	Incident response	Incident response comprises incident response policy and procedures, incident response training, incident response testing and exercises, incident handling and monitoring, incident reporting, and plan.
Operational	Maintenance	Maintenance comprises system maintenance policy and procedures, controlled maintenance, maintenance tools, non-local maintenance, maintenance personnel, and timely maintenance.
Operational	Media protection	Media protection comprises media protection policy and procedures, media access, media marking, media storage, media transport, and media sanitization.
Operational	Personnel security	Personnel security comprises personnel security policy and procedures, position categorization, personnel screening, Access Agreements, third-party personnel security, and personnel sanctions
Operational	Physical and environmental protection	Physical and environmental protection comprises physical and environmental protection policy and procedures, physical access authorizations and control, access control for transmission medium and for output devices, monitoring physical access, visitor control and access records, location of information system components, and information leakage.
Management	Planning	Planning comprises security planning policy and procedures, system security plan, rules of behavior, privacy impact assessment, and security-related activity planning
Management	Program management	Program management family comprises information security program plan, information security resources, plan of action and milestones process, information security measures of performance, risk management strategy, security authorization process, and mission/business process definition.
Management	Risk assessment	Risk assessment comprises risk assessment policy and procedures, security categorization, risk assessment, risk assessment update, and vulnerability scanning.
Management	Security assessment and authorization	Security assessment and authorization family comprises risk assessment policy and procedures, security categorization risk assessment, and vulnerability scanning.
Technical	System and communications protection	System and communications protection family comprises application partitioning, security function isolation, denial of service protection, transmission integrity and confidentiality, trusted path, cryptographic key establishment and management, information in shared resources, use of cryptography, public access protections, public key infrastructure certificates, operating system-independent applications, virtualization techniques, and transmission preparation integrity.
Operational	System and information integrity	System and information integrity family comprises system and information integrity policy and procedures, malicious code protection, information system monitoring, security alerts-advisories-directives, security functionality verification, software and information integrity, spam protection, information input restrictions and validation, Error Handling, information output handling and retention, and predictable failure prevention.
Management	System and services acquisition	System and services acquisition family comprises system and services acquisition policy and procedures, information system documentation, software usage restrictions, user-installed software, security engineering principles, external information system services, developer configuration management and security testing, supply chain protection, and critical information system components.

Appendix B. Entity Attack

Table B.1
Types of attacks.

Type	Attack	Description
Active	Brute force	Brute force attack consists of accessing sensitive data through attempting multiple combinations for cracking passwords, API keys, and SSH logins. The presented list of vulnerabilities shows that the communication with another controller could be compromised by this type of attack. Therefore, unauthorized communication can be exploited.
Active	Denial of service	Denial of service consists of prevent authorized access to resources or the delaying of time-critical operations. The exploited causal factors are Missing Communication, Missing Feedback, Missing Input, Missing Control Action, and Missing External Information.
Active	Disruption	Disruption could cause the general system or major application to be inoperable for an unacceptable length of time. The disruption attack exploits Incomplete Process Model, Inconsistent Process Model, Inadequate Control Algorithm, and Measurement Inaccuracy.
Active	Spoofing	Spoofing attack consists of an adversary impersonates another device or user during communication in order to launch attacks against the system. Causal factors raised by spoofing attack are Inadequate Feedback, Unauthorized Communication, Wrong Communication, Wrong Input, Wrong External Information, Malformed Input, Malformed Control Action, Malformed External Information, Malformed Feedback, and Malformed operation.
Active	System modification	System modification attack modifies the system settings for malicious purposes, which may exploit causal factors such as incorrect process model, Inconsistent Process Model, Wrong External Information, Wrong Input, Wrong Communication, and Measurement Inaccuracy.
Passive	Eavesdropping	Eavesdropping attack is when an attacker listens passively to a communication to capture information which can be used in a subsequent active attack to masquerade as the claimant. It leads to the causal factor unauthorized communication.
Passive	System mapping	A system mapping attack inspections the potential points of exploit on a system.

Appendix C. Screenshots of the STAMP-based Ontology Tool

This appendix shows the screenshots of SOT to record the causal and security scenarios.
Figs. C.1–C.3.

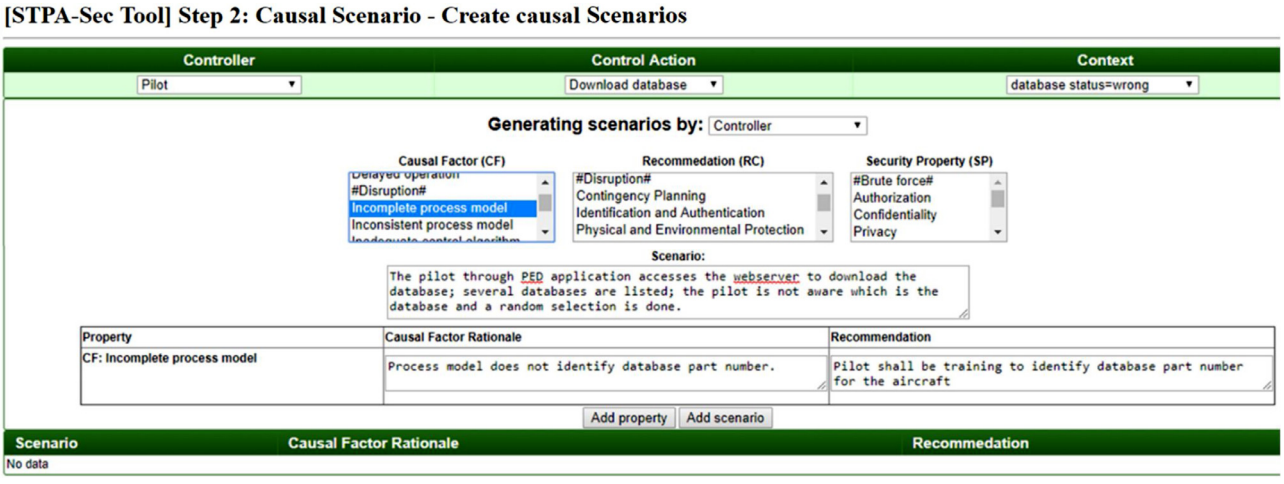

Fig. C.1. Causal Factors, Recommendations and Security properties by Controller.

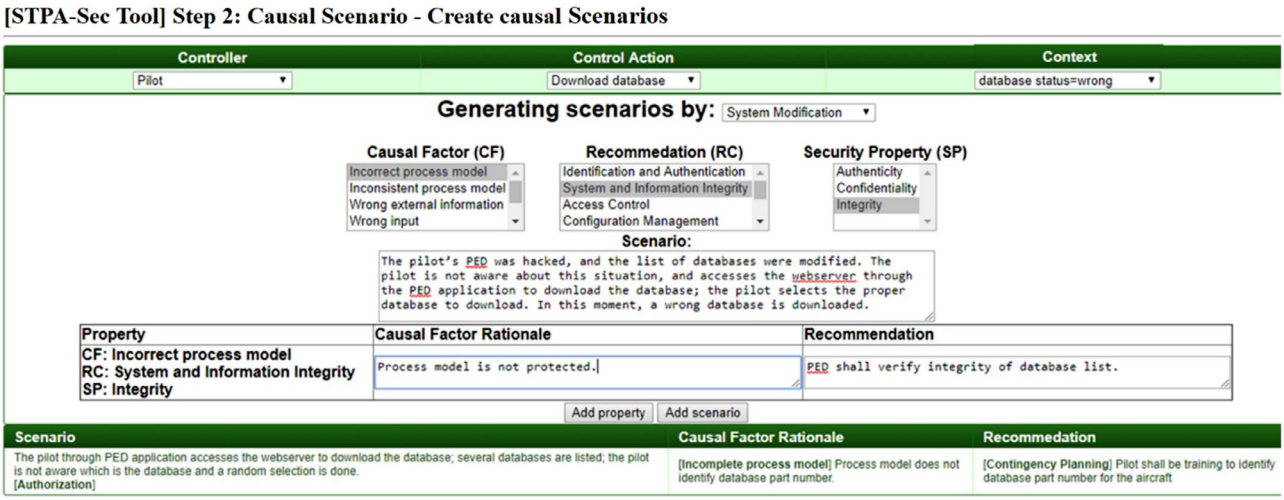

Fig. C.2. Causal Factors, Recommendations and Security properties by System Modification.

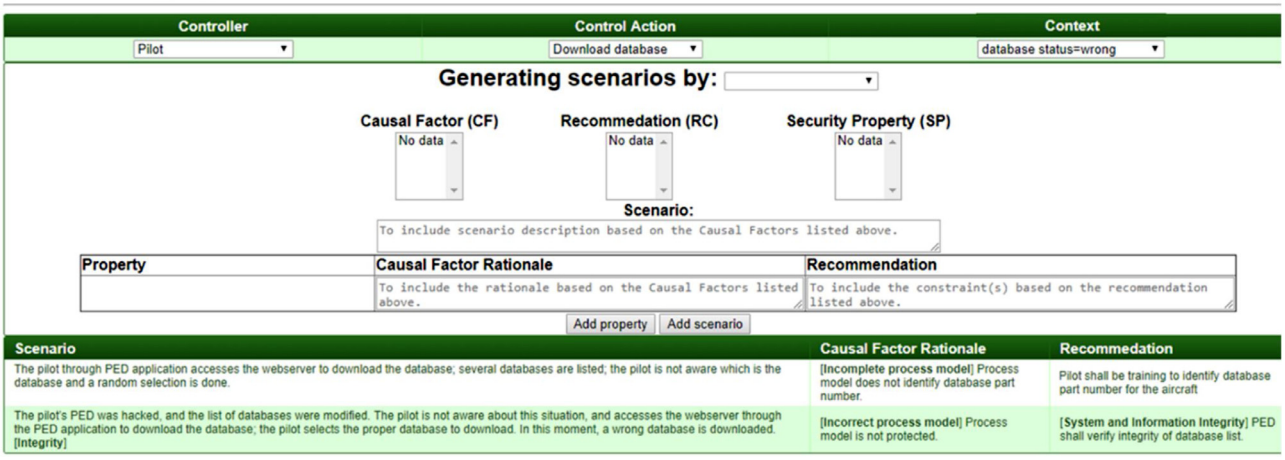

Fig. C.3. List of created causal scenarios.

References

[1] Bloomfield R, Vargas AP. Using ontologies to support model-based exploration of the dependencies between causes and consequences of hazards. In: Proceedings of the international conference on knowledge engineering and ontology development; 2015 http://openaccess.city.ac.uk/12801/.

[2] Ebrahimipour V, Yacout S. Ontology modeling in physical asset integrity management - FMEA, HAZID, and ontologies. Springer; 2011. p. 2–15. doi:10.1109/ETFA.2011.6058981.

[3] Elahi G, Yu E, Zannone N. A modeling ontology for integrating vulnerabilities into security requirements conceptual foundations. In: Conceptual modeling, 5829. Springer; 2009. p. 99–114. doi:10.1007/978-3-642-04840-1_10.

[4] Goluch G, Ekelhart A, Fenz S, Jakoubi S, Tjoa S, Muck T. Integration of an ontological information security concept in risk-aware business process management. In: Proceedings of the forty-first annual Hawaii international conference on system sciences; 2008.

[5] Heerden RP, Irwin B, Burke ID. Classifying network attack scenarios using an ontology. In: Proceedings of the ICIW seventh international conference on information warfare and security, Seattle, USA. University of Washington; 2012 http://hdl.handle.net/10962/d1009326.

[6] Herndon AA. Flight management computer (FMC) navigation database capacity. In: Proceedings of the integrated communications navigation and surveillance (ICNS); 2012.

[7] Herzog A, Shahmehri N, Duma C. An ontology of information security. J. Techn. Appl. Adv. Inf. Privacy Secur. 2007:278–301 IGI Global. doi:10.4018/jisp.2007100101.

[8] Massacci F, Mylopoulos J, Paci F, Tun TT, Yu Y. An extended ontology for security requirements. In: Proceedings of the CAiSE: advanced information systems engineering workshops; 2011. p. 622–36. doi:10.1007/978-3-642-22056-2_64.

[9] Rosa FF, Bonacin R, Jino M. A survey of security assessment ontologies. Recent advances in information systems and technologies, 569. Springer; 2017. World CIST. Advances in Intelligent Systems and Computing. doi:10.1007/978-3-319-56535-4_17.

[10] Rosa FF, Jino M, Bonacin R. Towards an ontology of security assessment: a core model proposal. In: Information technology - new generations. Advances in Intelligent systems and computing, 738. Springer; 2018. p. 75–80. doi:10.1007/978-3-319-77028-4_12.

[11] Souag A, Salinesi C, Mazo R, Comyn-Wattiau I. A security ontology for security requirements elicitation. In: Engineering secure software and systems (ESSoS). In: Lecture Notes in Computer Science, 8978. Springer; 2015. p. 157–77. doi:10.1007/978-3-319-15618-7_13.

[12] Souza FGR, Pereira DP, Pagliares RM, Nadjm-Tehrani S, Hirata CM. WebSTAMP: a web application for STPA/ STPA-Sec. In: Proceedings of the international Cross-industry Safety Conference (ICSC) - European STAMP Workshop & Conference (ESWC), 273; 2018. doi:10.1051/matecconf/201927302010.

[13] Vasilevskaya M. Security in embedded systems - a model-based approach with risk metrics PhD Thesis. Linkoping, Sweden: Department of Computer and Information Science Linkoping University; 2015.

[14] Young W, Leveson NG. An integrated approach to safety and security based on systems theory. Commun. ACM 2014;57(2):31–5. doi:10.1145/2556938.

[15] Zhou J, Hanninen K, Lundqvist K, Provenzano L. An ontological interpretation of the hazard concept for safety-critical systems. In: Proceedings of the twenty-seventh European safety and reliability conference. doi:10.1201/9781315210469-157.

[16] Zhou J, Hanninen K, Lundqvist K, Provenzano L. An ontological approach to identify the causes of hazards for safety-critical systems. In: Proceedings of the second international conference on system reliability and safety (ICSRS). doi:10.1109/ICSRS.2017.8272856.

[17] RTCA 2014, Airworthiness Security Process Specification. Radio Technical Commission for Aeronautics (RTCA), ED-202A / DO-326A, Issued 08-06-14, Prepared by SC-216, https://standards.globalspec.com/std/9869201/rtca-do-326A.
[18] Ross R, McEvilley M, Oren J. NIST Special Publication 800-160: Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems, In National Institute of Standards and Technology - U.S. Department of Commernce, November 2016, https://csrc.nist.gov/publications/detail/sp/800-160/vol-1/final.
[19] National Vulnerability Database, NIST Special Publication 800-53: Security Controls and Assessment Procedures for Federal Information Systems and Organizations, In National Institute of Standards and Technology – U.S. Department of Commerce, https://nvd.nist.gov/800-53.
[20] Leveson N. Engineering a safer world: Systems thinking applied to safety. Mit Press; 2011.