international journal of critical infrastructure protection 43 (2023) 100642

Contents lists available at ScienceDirect

International Journal of Critical Infrastructure Protection

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

A systematic co-engineering of safety and security analysis in requirements engineering process

Sejin Jung [a], Junbeom Yoo [b,*,1], Sam Malek [c]

[a] Chinju National University of Education, Republic of Korea
[b] Konkuk University, Republic of Korea
[c] University of California, Irvine, United States of America

ARTICLE INFO

Keywords:
Co-engineering
Safety analysis
Security analysis
Goal-tree analysis
Requirements engineering
Safety-critical systems

ABSTRACT

Co-engineering safety and security is increasingly important in safety-critical systems as more diverse interacting functions are implemented in software. Many studies have tried to perform safety and security analyses in unified or in parallel. While the unified approach requires more complex analysis with new delicate methods, the parallel needs further improvement on additional integration activity for harmonizing safety and security analyses results. This paper tries to improve the harmonization activity seamlessly and systematically in typical requirements engineering process for safety-critical systems. It encompasses both requirements elicitation and analysis as well as safety and security analyses, regardless of which analysis techniques are used. The paper suggests performing an appropriate safety analysis first to derive safety requirements as summary information. It then performs goal-tree analysis to refine the high-level safety requirements into lower-level ones, from which any security analysis can work on to derive security requirements. Another goal-tree analysis then tries to refine the high-level security requirements into specific functional ones too, and it ends the analysis activity in a cycle of requirements engineering process. The sequence of safety analysis, goal-tree refinement, security analysis and another goal-tree refinement is seamlessly iterated in the process of requirements engineering, where any conflict of requirements will have an opportunity to be resolved. Our case study of a simplified UAV example uses STPA and STRIDE techniques for safety and security analysis respectively, and shows that the proposed approach is fully applicable up to industrial cases.

1. Introduction

Co-engineering safety and security [1,2] is becoming increasingly important to even traditional safety-critical systems, as they are trying to integrate security provisions into systems and architectures, which are fundamentally designed for safety, to avoid potential conflicts between safety and security provisions. It also aims to aid the identification and the leveraging of the potential synergies between safety and security. For nuclear power plants' I&C (Instrumentation and Control) systems, the safety and security coordinating framework IEC 62859:2016 [3] was proposed, based on functional safety standards such as IEC 61508:2010 [4], IEC 60880:2006 [5] and IEC 62138:2018 [6]. Many studies in the field of CPS (Cyber Physical System) [7], SoS (System of Systems) [8] and control systems [9] also have tried to co-engineering safety and security analysis in combination, *i.e.,* in unified or in parallel, as well surveyed in [10].

The unified approach proposes joint analysis of safety and security and their interdependencies in one method, but requires more complex analysis with new delicate notations. The parallel, on the other hand, performs familiar safety and security analyses individually as usual, but requires additional integration activity for harmonizing safety and security analyses results, which is often expected with further improvement [10]. The latter might reduce the insight of the analysis leading to unbalanced results, while the former provides more rigorous results, with better understanding of potential conflicts between safety and security [11].

Traditional safety-critical systems such as nuclear power plants, airplanes, vehicles and trains still have required more strict and thorough demonstration of safety in accordance with functional safety standards (*e.g.,* IEC 60880, ISO 26262, DO 178B/C, EN 50128) for commercial operation. Engineers still find it comfortable to derive security requirements additionally (*i.e.,* in parallel) and analyze the interrelationship between safety and security while maintaining the safety life-cycle of functional safety standards, as investigated by [12]. This paper aims to improve the harmonization activity of safety and security in

* Correspondence to: Konkuk University, Republic of Korea.
E-mail address: jbyoo@konkuk.ac.kr (J. Yoo).
[1] He was a visiting scholar to University of California, Irvine, USA.

https://doi.org/10.1016/j.ijcip.2023.100642
Received 26 May 2023; Received in revised form 10 September 2023; Accepted 16 November 2023
Available online 18 November 2023

typical requirements engineering process for safety-critical systems, while keeping using typical software development processes based on functional safety standards and HARA (Hazard Analysis and Risk Assessment) techniques. It encompasses both requirements engineering (*i.e.,* elicitation and analysis) as well as hazard and vulnerability analyses, regardless of which analysis techniques are used.

This paper tries to perform safety and security analysis together, *i.e.,* in parallel, without introducing new notations but considering their interdpendencies seamlessly in a typical process of requirements engineering [13] for safety-critical systems. At the requirements analysis phase in software development life-cycles (SDLC) [14], requirements engineering activities such as requirements '*elicitation*,' '*analysis*,' '*specification*' and '*validation*' are performed iteratively up to finish an SRS (Software Requirements Specification) [15]. We try to integrate any safety and security analyses into the activities of requirements engineering, seamlessly and systematically.

This paper suggests performing an appropriate safety analysis first as part of the '*elicitation*' to derive safety requirements in accordance with functional safety standards. It then performs '*Goal-Tree Analysis*' [16] as an '*analysis*' technique to refine the high-level safety requirements into lower-level ones. From the specific functional requirements for safety, any security analysis technique can work on to derive security requirements. Most security analysis techniques are not easy to link to safety analysis results directly, because the security analysis often requires more detailed information than safety analysis produces. Another goal-tree analysis then tries to refine the high-level security requirements into specific functional ones. The sequence of safety analysis, goal-tree refinement, security analysis and another goal-tree refinement is seamlessly iterated in the process of requirements engineering. Of course, the refined functional requirements for security may cause a conflict with the functional ones for safety. However, it will have a chance to be resolved in the iterative process of requirements engineering. Requirements also typically keep changing and affecting the safety and security requirements analyzed so far, and it will also have to be resolved naturally in the cyclic process of requirements engineering, as we are currently working on now.

This paper performs a case study with a simplified UAV example, applying STPA and STRIDE as safety and security analysis techniques, respectively. It shows that the proposed approach is fully applicable to industrial cases with various safety and security analysis techniques. The remainder of the paper is organized as follows. Section 2 summarizes the related work trying to analyze safety and security together. Section 3 overviews the STPA analysis technique, the goal-tree analysis and the STRIDE security model to aid understanding the case study. Section 4 describes the overall co-engineering process of safety and security analysis in requirements engineering process. Section 5 presents the case study with a simplified UAV example, and the paper concludes with future work in Section 6.

2. Related work

Many studies have addressed the integration of safety and security analysis for various application domains such as CPS (Cyber Physical System) [7], SoS (System of Systems) [8] and control systems [9]. Comprehensive reviews on safety and security analysis techniques that can be integrated in unified or in parallel are also well provided by [17,18]. A systematic literature review [10] classifies the safety and security co-analysis work based on their focus, such as '***Combined safety and security approach***' vs. '*Security informed safety approach*' and '***Parallel***' vs. '*Unified*.' Researches are also classified by the SDLC (Software Development Life-Cycle) stage that they address, such as '*RE only*,' '*HA and TA only*' and '***RE, HA and TA***.[2]' The co-engineering of safety and security in requirement engineering process for safety-critical systems, which this paper proposes, fits to the category of "*Combined safety and security approach in parallel encompassing requirements engineering, hazard analysis and threat analysis altogether*" technique.

From the perspective of traditional safety-critical systems, studies can also be categorized based on two accident models [19] such as '*chain-of-event*' and '*STAMP (Systems-Theoretic Accident Model and Processes)*.' Based on the '*chain-of-event*' accident model, many traditional safety analysis techniques [20] have been extended to new ones which can also analyze security. FMVEA (Failure Mode, Vulnerabilities and Effects Analysis) [21], SeCFT (Security Enhanced Component Fault Trees) [22], SAHARA (A Security-Aware Hazard Analysis and Risk Analysis) [23], CHASSIS (Combined Harm Assessment of Safety and Security for Information Systems) [24] and many others [25,26] all try to deal with security as well as safety through one new integrated model.

Several studies are also based on the '*STAMP*' accident model [27] and its safety analysis technique, STPA (Systems-Theoretic Process Analysis) [28]. While STPA-Sec [29] is a widely-used security analysis technique which has shown a number of successful applications [30–32], it only analyzes security features. It does not extend the STPA-based safety analysis up to security, but adopts a similar analysis process from STPA to analyze security. [33] proposes a GSN (Goal-Structuring Notation) pattern to derive integrated safety and security requirements. [34] builds a goal-tree which tries to connect to each other, *i.e.,* all models analyzing safety and security individually. It does not perform the goal-tree analysis like this paper but uses the notation of goal-trees. [35] tries to integrate STPA with '*Six Step Model*' for security analysis. [36] proposes an annotated control graph to cope with safety and security together. On the other hand, [37] presents a combined analysis method for safety and security called STPA-SafeSec based on STPA and STPA-Sec, and [38] also proposes an approach combining STPA-Sec and FMVEA. [39] tries to combine STPA and Secure Tropos in parallel while trying to weave two analyses results in sequence. We use STPA and STRIDE analyses techniques in the case study, but any familiar analysis technique can be used in parallel without interfering others in cyclic requirements engineering process.

Some studies also try to extend security analysis techniques into safety as well surveyed in [10]. [40] extends a TVRA (Threat Vulnerability and Risk Assessment) technique with SIL (Safety Integrity Level) from the functional safety standard IEC 61508. [41] tries to extend the NIST 800–30 methodology to consider safety aspects. While most approaches try to use new notations and models to aid analyzing security features as well as safety, this paper tries to achieve the same goal without introducing new notations nor extending the existing ones. We can also incorporate any safety and security analysis techniques in familiar requirements engineering processes. The case study in the paper uses STPA for safety analysis and STRIDE for security without extension, and of course any other analysis techniques can be incorporated, too.

3. Background

3.1. STPA

STPA (Systems-Theoretic Process Analysis) [28] is a safety analysis technique based on the STAMP (Systems-Theoretic Accident Model and Processes) accident model [27]. The system is viewed as a hierarchical structure that higher-level components control lower-level components which feed back to higher ones according to the '*control structure*' model. The STPA focuses on identifying hazardous controls, *i.e.,* UCAs (Unsafe Control Actions) between the controllers and controlled components as described in ⟨Fig. 1⟩.

UCAs are classified in 4 types such as '*not providing causes hazard*,' '*providing causes hazard*,' '*too early, too late, or out of order*,' and '*stopped too soon or applied too long*.' Safety analysts try to identify control

[2] RE: Requirements Engineering, HA: Hazard Analysis (Safety), TA: Threat Analysis (Security)

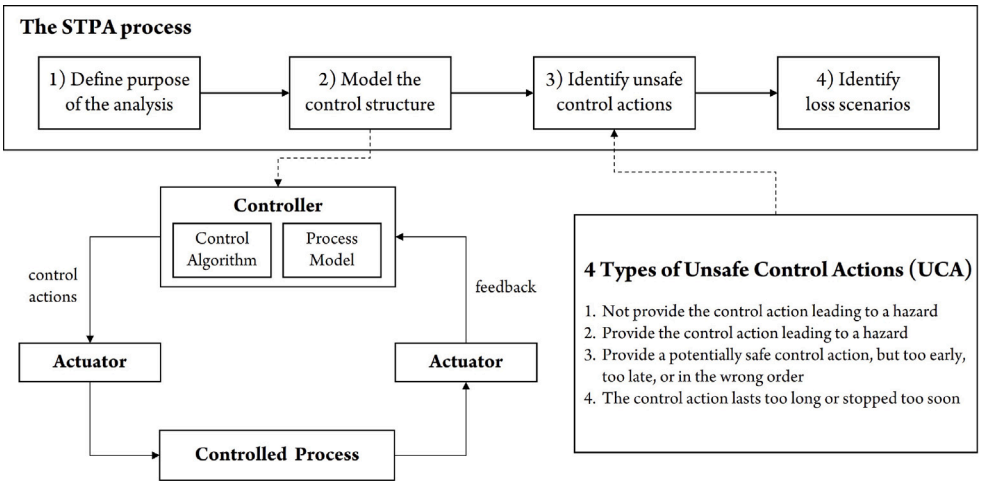

Fig. 1. The typical analysis process of STPA.

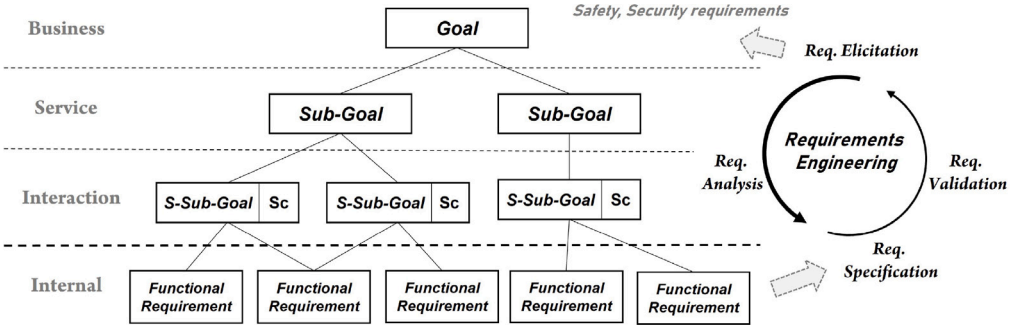

Fig. 2. A goal-tree analysis in requirement engineering process.

commands which can result in UCA, and then analyze L/CS (Loss/Cause Scenarios) on the control structure, which can contribute to the losses. Safety experts have to look them closely one by one to see if they can cause real accidents or not. Safety requirements can be formulated from L/CS, but it is out of the scope of the STPA analysis itself.

3.2. Goal-tree analysis in requirements engineering

Requirements engineering [13] is a structured set of activities to identify requirements and organize them into SRS (Software Requirements Specification) [15]. It includes '*requirements elicitation*' from stakeholders to discover user requirements and '*requirements analysis*' with requirements models to specify system requirements which are required to satisfy the user requirements. It also encompasses '*requirements specification and validation*' to organize all types of requirements (*e.g.,* user, system, functional, non-functional and quality requirements) into a SRS and check their validity and C&C (Completeness and Consistency). All activities are iterated continuously until a desired level of SRS is completed, depending on the SDLC (Software Development Life-Cycle) [14] used, as described on the right side of ⟨Fig. 2⟩.

A *goal-tree* is a general form of mind mapping to help visually to break down a large high-level goal into smaller sub-goals. Goal-oriented requirements engineering [16,42] uses it to transform high-level user requirements (goals) into low-level functional requirements as depicted in ⟨Fig. 2⟩. It corresponds to the '*Req. Analysis*' activity in the requirements engineering process.

The goal at the business level in a goal-tree is derived from the '*Req. Elicitation*' activity and often includes quality attributes such as *safety* and *security* [43]. Engineers then keep refining the goal into lower-level goals subsequently. The refined goals at the interaction level can have scenarios (Sc), like '*Use-Case*' in UML, which are composed of specific functional requirements at the internal level.

This paper performs the goal-tree analysis twice in a cycle, for safety requirements and security requirements, respectively. It starts the goal-tree analysis with the *safety requirements* formulated from the results of safety analysis. Upon the specific functional requirements at the internal level, this paper also suggests applying security analysis techniques to analyze threats and derive security requirements. Another goal-tree analysis starts with the security requirements to refine them into more specific functional requirements to satisfy the security requirements.

3.3. STRIDE threat model

STRIDE [44] is a model for identifying computer security threats by Microsoft. It provides 6 categories of security threats, such as *spoofing*, *tampering*, *repudiation*, *information disclosure*, *denial of service* and *elevation of privilege*. The STRIDE is used to help reason and find threats to a system. It can be also used in conjunction with a model of the target system, *e.g.,* control structures in STPA. It is often used by security experts to help answer questions [45] such as

- *How can an attacker change the authentication data?*
- *What is the impact if an attacker can read the user profile data?*
- *What happens if access is denied to the user profile database?*

As a widely-used threat model. STRIDE is often integrated with safety analysis techniques like FMEA as proposed by [21,23,25]. [46,47] also presents a threat modeling framework for CPS using STRIDE, considering system security at the component level. The case study in the paper uses the STRIDE model as a security analysis technique, which are widely-used for general application, but any other technique can also be used without modification.

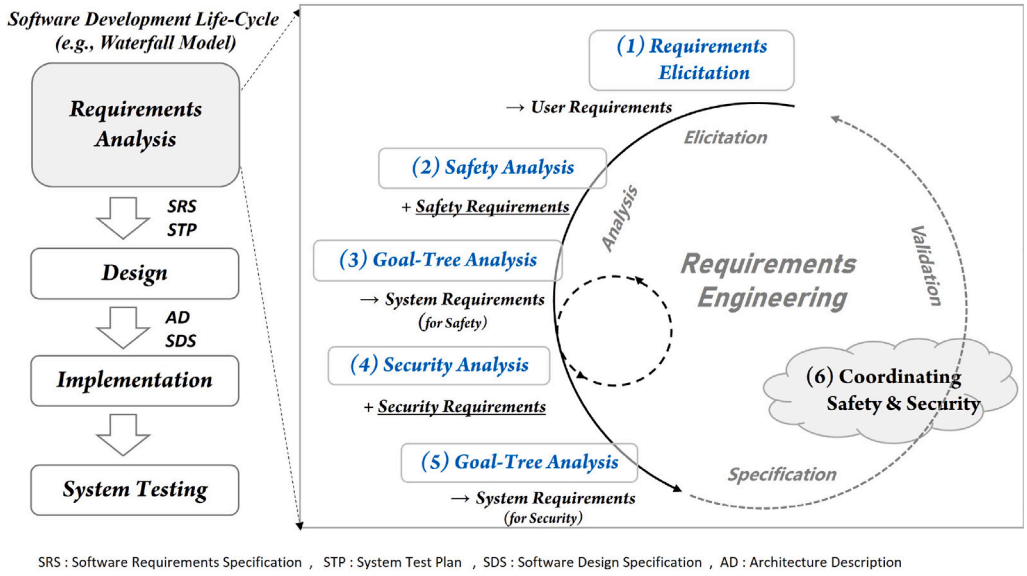

Fig. 3. An overview of the co-engineering process of safety and security analysis.

4. The co-engineering process of safety and security analysis

The co-engineering process of analyzing safety and security together is naturally embedded in the '*requirements engineering process*' of a '*software development life-cycle*' as depicted in ⟨Fig. 3⟩. It does not require any new notation or technique other than the safety and security analysis techniques, which have been widely used and familiar with. It suggests the sequence of requirements elicitation, safety analysis, goal-tree refinement, security analysis and another goal-tree refinement in cyclic iterations of requirements engineering process within a software development life-cycle of safety-critical systems.

- **(1) *Requirements Elicitation*** It is the typical first step in requirements engineering process. It tries to elicit, *i.e.,* identify or discovery, from stakeholders high-level '*user requirements*' through appropriate elicitation techniques such as requirements workshop, brainstorming, role playing, survey, interview, prototyping and etc. We do not assume that particular techniques are used, but assume that '*requirements negotiation*' [48] results in an organized set of user requirements.
- **(2) *Safety Analysis*** Any '*safety analysis*' technique is applicable to the '*user requirements*' and yields a number of valuable analytic results that both safety analysts and system engineers need to take a closer look at.
  - ***Formulating Safety Requirements*** The co-engineering process first suggests that the vast safety analysis results be formulated into a more refined form, '*safety requirements*.' It is the same way that safety requirements are derived from HARA (Hazard Analysis and Risk Assessment) in functional safety standards. It prevents security analysts from being caught up in a lot of unfeasible security issues that are far from the actual implementation of the system, later but soon in the co-engineering process.

  The safety requirements act as '*summary information*' for system engineers to achieve to improve the system safety through implementing required functionalities. The specific functional, *i.e.,* system requirements, will be derived at the next step through goal-tree analysis.
- **(3) *Goal-Tree Analysis*** The '*goal-tree analysis*' then refines the user-level safety requirements into lower-level '*system requirements*,' which can achieve the safety requirements in the system under consideration. While requirement models such as DFD (Data Flow Diagram) and Use-Cases are typically used to aid refining, they are limited to refining functional requirements. The '*goal-tree analysis*' is more effective in refining both functional and non-functional (*e.g., safety* and *security*) requirements together at the user-level. The goal-tree analysis in the co-engineering process starts from the safety requirements formulated well and keeps refining into lower-level functional requirements at the system level. These specific functional requirements serve as a stepping stone to connect the safety analysis to security analysis performed next.
- **(4) *Security Analysis*** From the refined functional requirements for safety at the system level, security experts are finally able to encounter valuable information to predict the use of a draft software architecture, network protocols, specific hardware and software, or safety solutions in consideration. Security experts now can perform any '*security analysis*' based on the specific information. They will identify some functional requirements called '*security concerns*' which might degrade the system security, even if they were devised to improve the system's safety.
  - ***Formulating Security Requirements*** Security experts can derive more specific threat scenarios and formulate '*security requirements*' to resolve the threat scenarios. The security requirements now are supposed to improve the system security while preserving the system safety, simultaneously.
- **(5) *Goal-Tree Analysis*** Another goal-tree analysis starts from the security requirements and keeps refining into detailed functional requirements to achieve the security ones. Security analysts as well as system architects often get useful hints from '*security tactics*' [49] to achieve the security goals. The refined functional requirements for security might cause a conflict with the ones for safety, but it will be analyzed in the next step.
- **(6) *Coordinating Safety and Security*** The entire process is conducted through cooperation between safety and security experts. Iterations of requirements modification, refinement and re-analysis of safety and security are always accompanied by typical requirements engineering. A number of security requirements will be derived and may have negative impacts on other safety requirements that they have not analyzed with. Some functional requirements for safety may conflict with some functional requirements for security, and vice versa. A functional requirement

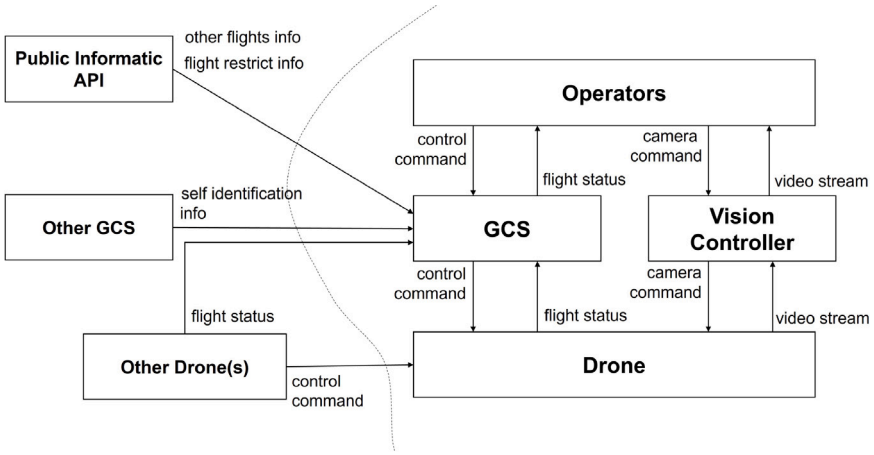

Fig. 4. A high-level view of the control structure for the UAV example.

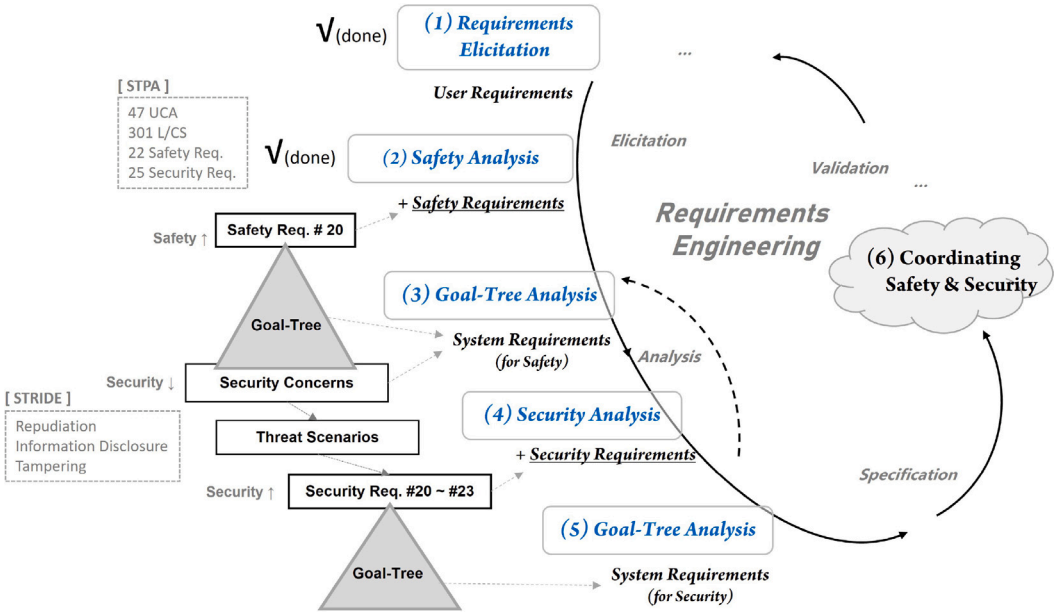

Fig. 5. An overview of the case study with STPA and STRIDE.

for security also may cause a conflict with other high-level safety requirements from which that was not derived. A systematic analysis on the '*coordinating safety and security*' will be required at the requirements specification and validation steps during the requirements engineering process. A more detailed consideration on the coordination comes later, but we are working on it now and a sketch for the coordination will be introduced.

5. The case study: A simplified UAV

5.1. An overview of the case study

The case study aims to demonstrate how the co-engineering process works seamlessly in the typical process of requirements engineering with widely-used safety and security analysis techniques. STPA is used for safety analysis technique and STRIDE for security. The target system is a simplified UAV (Unmanned Aerial Vehicle) for fire surveillance, which our previous work [30] had used to derive safety requirements from STPA and security requirements from STPA-Sec, respectively. We had expected the two techniques to share a lot of analytical information and processes, but few other than '*control structures*' roughly shown in ⟨Fig. 4⟩. It motivated us to propose the co-engineering process for safety and security analysis in the paper.

As a typical UAV, the example includes 4 participants such as an operator, a GCS (Ground Control System), a vision controller, and drone(s), as shown in ⟨Fig. 4⟩. (More refined ones were used for the STPA analysis [30].) There are 4 basic elements attached to control actions and feedbacks such as '*control command*' and '*flight status*.' On the left, there are three elements outside the system under consideration. They can send control actions to the target system.

⟨Fig. 5⟩ overviews the whole process of the case study, in accordance with the proposed co-engineering process depicted in ⟨Fig. 3⟩. STPA and the STRIDE analysis are used and it describes the security analysis steps in more detail to aid understanding. The goal-tree analysis is also used to refine the high-level safety and security requirements into low-level functional requirements in requirements engineering process. The figure simplifies the repeated application of goal-tree analysis in sequence for convenience.

The case study begins by assuming that the '*(1) Requirements Elicitation*' and '*(2) Safety Analysis*' had been completed by the previous work. It had resulted in 47 UCA (Unsafe Control Actions), 301 L/CS (Loss/Cause Scenarios), 22 safety requirements and 25 security requirements. While the details of the analyses are out of the scope of the

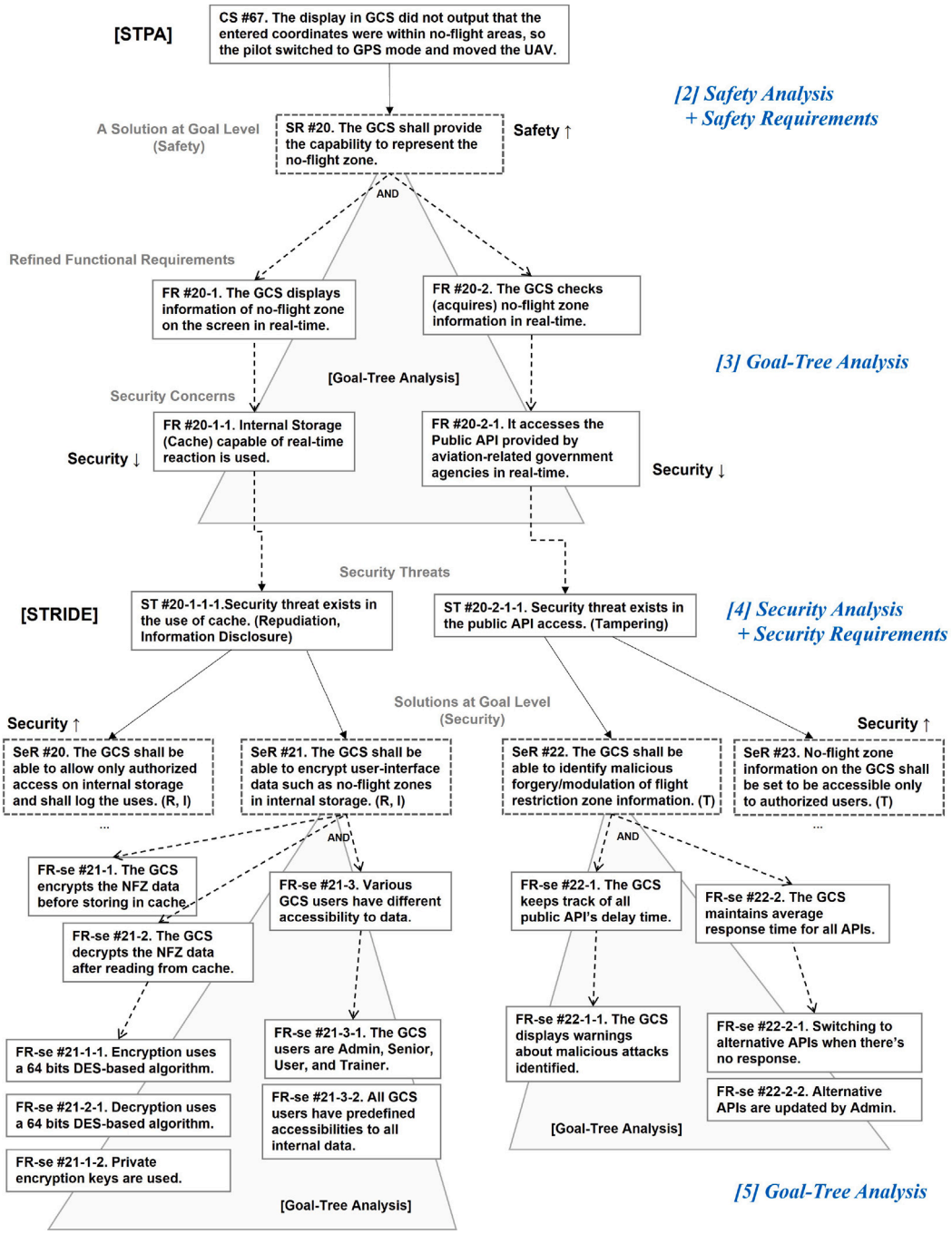

Fig. 6. An overview of the case study starting from the safety requirement #20.

paper, the case study shows how one safety requirement (No. 20) can be refined into specific functional requirements through a goal-tree analysis. And seamlessly a STRIDE analysis results in 4 security requirements, which are expected to improve safety and security simultaneously. Our previous work with STPA-Sec had identified only a half. It also refines the security requirements into specific functional requirements by performing another round of goal-tree analysis, and it will help us get insights into our future work, '*(6) Coordinating Safety and Security*'.

5.2. The case study results

The case study conducts the 5 steps summarized in ⟨Fig. 5⟩ in order. The first two steps had been completed in our previous work, which resulted in 47 UCA, 301 L/CS, 22 safety requirements and 25 security requirements. The case study used some of the results and shows how one safety requirement (No. 20) is refined and analyzed to derive 4 security requirements (No. 20 ~ No. 23) and their specific functional requirements, seamlessly in the process of requirements engineering, as shown in ⟨Fig. 6⟩.

(1) *Requirements Elicitation* We assume completion.

(2) *Safety Analysis* We assume that the second step was also completed. The STPA analysis was performed and all safety requirements were derived. Among the 22 safety requirements and associated UCAs and L/CS, the case study chose the ***safety requirement*** #20 to derive related security requirements through the proposed co-engineering process. In detail, UCA #9 and L/CS #67 are concerned with deriving the SR #20, as described in the table below. The upper part of ⟨Fig. 6⟩ shows

the safety requirement (SR #20) starting the whole co-engineering process.

UCA #9	The Pilot entered a GPS coordinate within the no-flight area and switched to GPS mode.
L/CS #67	The display in GCS did not output that the entered coordinates were within no-flight areas, so the pilot switched to GPS mode and moved the UAV.
SR #20	The GCS shall provide the capability to represent the no-flight zone.

(3) *Goal-Tree Analysis* Starting from the safety requirement (SR #20), we conduct the *goal-tree analysis* to refine it into 4 subsequent ***functional requirements*** as depicted in the triangle tree at the upper part of ⟨Fig. 6⟩. It is a typical requirements analysis activity that has nothing to do with specific safety analysis techniques. They are considered to be the functional requirements at the interaction and internal levels in ⟨Fig. 2⟩, and will be used to realize the safety requirement.

FR #20-1	The GCS displays information of no-flight zone on the screen in real-time.
FR #20-2	The GCS checks (acquires) no-flight zone information in real-time.
FR #20-1-1	Internal Storage (Cache) capable of real-time reaction is used.
FR #20-2-1	It accesses the Public API provided by aviation-related government agencies in real-time.

(4) *Security Analysis* Security experts (actually, graduate students in the case study) identify two ***security concerns*** in the refined functional requirements for safety. The use of internal storage like cache (FR #20-1-1) and public APIs (FR #20-2-1) may provide the cause of security attacks. At this point, security experts are able to notice possible security threats which could degrade the security of the system under consideration. For each security concern, security experts now try to identify possible and specific ***security threats*** with support of ***threat models***. The followings are the result of the STRIDE model's help.

ST #20-1-1-1	Security threat exists in the use of cache. (Repudiation, Information Disclosure)
ST #20-2-1-1	Security threat exists in the public API access. (Tampering)

For the functional requirement (FR #20-1-1), security threat (ST #20-1-1-1) is anticipated as the system is going to use a cache storage which could suffer from the attacks in the categories of *repudiation* and *information disclosure*. On the other hand, security threat (ST #20-2-1-1) in the category of *tampering* was identified for (FR #20-2-1), since the use of public APIs on the cloud network may cause the no-flight-zone information to be intercepted.

The last step is to formulate ***security requirements***, which can reduce, mitigate, or prevent the security threats. The security experts formulate 2 security requirements against each security threat (ST #20-1-1-1) and (ST #20-2-1-1), respectively. The two security requirements for (ST #20-1-1-1) were not devised from our previous work with STPA-Sec. Since the corresponding security threat scenario (ST #20-1-1-1) was only derivable from the specific functional requirements (FR #20-1-1) and (FR #20-1), which the goal-tree analysis refined and identified. These four security requirements all aim to improve the system's security, even if the safety requirement (SR #20) could degrade the system security during its subsequent implementation to improve the system safety.

SeR #20	The GCS shall be able to allow only authorized access on internal storage and shall log the uses. (R,I)
SeR #21	The GCS shall be able to encrypt user-interface data such as no-flight zones in internal storage. (R,I)
SeR #22	The GCS shall be able to identify malicious forgery/modulation of flight restriction zone information.(T)
SeR #23	No-flight zone information on the GCS shall be set to be accessible only to authorized users.(T)

(5) *Goal-Tree Analysis* Another round of goal-tree analysis now starts from the security requirements to derive specific functional requirements needed to achieve the security goals. The case study introduces the refinement of two security requirements due to the space constraints. For the security requirement (S*e*R #21), 3 and 4 functional requirements are refined subsequently. As the security requirement requires encryption for the no-flight zone information stored in an internal storage, specific requirements for the data encryption (FR-se #21-1) and decryption (FR-se #21-2) were refined. A 64 bits DES (Data Encryption Standard) is also chosen as the encryption algorithm (FR-se #21-1-1, FR-se #21-1-2). The DES-based encryption uses a unique private key for each user (FR-se #21-1-2). It led us to derive (FR-se #21-3) that varies accessibility to internal data according to user classification and subsequent ones (FR-se #21-3-1, FR-se #21-3-2).

FR-se #21-1	The GCS encrypts the NFZ data before storing in cache.
FR-se #21-2	The GCS decrypts the NFZ data after reading from cache.
FR-se #21-3	Various GCS users have different accessibility to data.
FR-se #21-1-1	Encryption uses a 64 bits DES-based algorithm.
FR-se #21-1-2	Private encryption keys are used.
FR-se #21-2-1	Decryption uses a 64 bits DES-based algorithm.
FR-se #21-3-1	The GCS users are Admin, Senior, User, and Trainer.
FR-se #21-3-2	All GCS users have predefined accessibilities to all internal data.

On the other hand, the security requirement(S*e*R #22) is refined in to 2 and 3 functional requirements subsequently. As the security requirement requires the GCS to detect malicious attacks on the API and data, specific requirements for detecting attacks on the API (FR-se #22-1, FR-se #22-2) are refined. More specific functional requirements (FR-se #22-1-1 ∼ 3) are also derived based on the security detection tactics [49] such as '*detect message delay*' and '*detect service denial (DoS)*.'

FR-se #22-1	The GCS keeps track of all public API's delay time.
FR-se #22-2	The GCS maintains average response time for all APIs.
FR-se #22-1-1	The GCS displays warnings about malicious attacks identified.
FR-se #22-2-1	Switching to alternative APIs when there's no response.
FR-se #22-2-2	Alternative APIs are updated by Admin.

It is worth to note that a systematic labeling and categorization of levels of requirements is a key to ensuring efficient traceability [50]. This paper used a traceability analysis tool, '*ARTracer* [51]' to manage them in an efficient and integrated manner, while the labels used in the case study were modified to aid understanding.

5.3. Evaluation

In order to demonstrate the validity and effectiveness of the proposed approach, we investigate the following two research questions and more consideration on our future research direction.

RQ1. ***Validity***. *Is it possible to derive security requirements directly from safety analysis results?*

RQ2. ***Effectiveness***. *Does the goal-tree analysis provide sufficient information for security analysis?*

- **RQ1.** ***Validity.*** *Is it possible to derive security requirements directly from safety analysis results?*
  It is typically not straightforward to derive *security concerns* and *threats* directly from the STPA analysis results such as UCA and L/CS and even from the summary information, *safety requirements*. ⟨Fig. 7⟩ illustrates an example that security experts try to derive security requirements directly from the same safety requirement (SR #20). They need to do conduct a causal analysis on the safety requirement using a security threat model like STRIDE, but have no idea how it will be realized (implemented) for now. There will be many ways to implement the safety requirement, and the use of cache (ST #20-1-1-1) and public APIs (ST #20-2-1-1) are just one of options that architects and system engineers can choose.
  Most studies that try to integrate safety and security analyses results together in unified or in parallel have replaced this lack of information with "*the experience and ability of security experts.*" Specific unusual mappings to link specific analysis results to the other often have attempted, and make it difficult to use familiar and comfortable analysis techniques. However, If a goal-tree analysis naturally refines the safety requirement into specific functional ones as the previous example in ⟨Fig. 6⟩, we can take a look at what options are available now and can fill the gap seamlessly in the process of requirements engineering. Even if they all will not be used later, we can devise all possible security requirements for them all.
  Of course, not all cases need the refinement through the goal-tree analysis. There are many cases that security threats are derivable directly from safety requirements by '*experienced*' security experts as illustrated in ⟨Fig. 8⟩. Security threats such as "*Security threat may exist in the battery information displayed on the controller. (Tampering)*" seems to be easily derivable from the safety requirement (SR #16) directly by experienced experts. Our previous work [30], which tried simultaneous derivation of safety and security requirements without any aids, showed that about 10% (2 out of 22) of safety requirements were the type that security concerns and threats are hard to be driven out directly. Additional 20% of new security requirements could have been derived from them.
  The requirements engineering to derive safety and security requirements is typically an iterative process. Security experts need to analyze new security threats whenever new safety requirements and their refined ones are analyzed. We will keep encountering safety requirements that are not straightforward to derive security directly at any time at any iteration. In addition, we also need to store and keep track of all information yielded in the iterative process of safety and security analysis.
- **RQ2.** ***Effectiveness.*** *Does the goal-tree analysis provide sufficient information for security analysis?*
  We are now in the early stage of software development, *i.e.,* '*requirements analysis*.' Activities of requirements engineering are performed iteratively to elicit, evaluate and finalize all high-level functional and non-functional requirements. The functional requirements which the goal-tree analysis refines from high-level safety requirements will be sufficient for deriving appropriate security concerns, threats and requirements. Since we are working in the early stage of software development, '*requirements analysis*.' More detailed functional requirements will be analyzed and designed in the next phase, '*design*.'
  In this stage of software development, requirements experts and architects often cooperate to find the most appropriate system architecture which can best satisfy the safety and security requirements altogether. Architects often use '*tactics*' [49], a set of widely-used system engineering techniques, to meet the system's quality attributes such as *security*. While system engineers conduct the goal-tree analysis to refine safety and security requirements into more detailed ones, architects also consider the possibility of applying appropriate tactics, and the decisions are reflected back in the refined requirements, iteratively.
  For example, ⟨Fig. 9⟩ shows the tactics for *security*, and we can find various engineering techniques to achieve a required level of security. In the example of the case study described in ⟨Fig. 6⟩, the refined functional requirement (FR #20-2) and its security concern (FR #20-2-1) look sufficient for architects to derive tactics such as '*Authorize Actors*' and '*Detect Message Delay*' to resolve the security threats (ST #20-2-1-1).
- ***Consideration*** *on conflicts between safety and security*
  We keep encountering various conflicts between requirements in the process of requirements engineering, and the resolution of the conflicts is one of the most important roles of this stage [52]. Most studies [10] have tried to resolve the conflicts at once through specific ways such as developing an integrated model or mapping one to the other, *i.e.,* indirectly. However, requirements engineering as well as safety and security analyses are not an once-in-a-lifetime activity but a structured set of activities, performing iteratively up to meet a predefined goal. It is important to embrace into the co-engineering process the fact that "*Requirements keeps changing, safety and security analysis results keep changing, and the conflicts between elicited requirements keep changing, altogether.*" Safety experts, security experts and software system architects altogether should resolve the conflicts in the cycles of requirements engineering.
  A *Goal-Tree Analysis* provides notations such as '+ −' to note the positive and negative relationships between requirements. The '+ −' notations in goal-trees are not an extension, and they can be used to visually identify the inter-relationships between high-level requirements (*safety* and *security*) and lower-level functional requirements to achieve them. We are currently focusing on how to systematically identify, analyze and resolve the conflicts in requirements engineering process seamlessly, *i.e.,* '*(6) Coordinating Safety and Security*' in ⟨Fig. 3⟩.

6. Conclusion and future work

This paper proposes a systematic co-engineering process of safety and security analysis, naturally embedded in requirements engineering process. Any familiar safety and security analysis techniques can be applied together without additional effort. Any analysis techniques can also be harmonized with requirements analysis activities, *i.e., the goal-tree analysis*, in typical requirements engineering process. The proposed approach considers requirements elicitation and analysis, safety analysis and security analysis altogether within the cyclic process of requirements engineering, and recognizes that requirements as well as analyses results keep changing. It well understands well that engineers in the field of safety-critical systems still find it comfortable to keep going on with the safety life-cycle of functional safety standards.

The paper suggests performing an appropriate safety analysis first to derive safety requirements as summary information. It then performs goal-tree analysis to refine the high-level safety requirements into lower-level functional requirements, from which any security analysis can work on to derive security requirements. Another goal-tree analysis

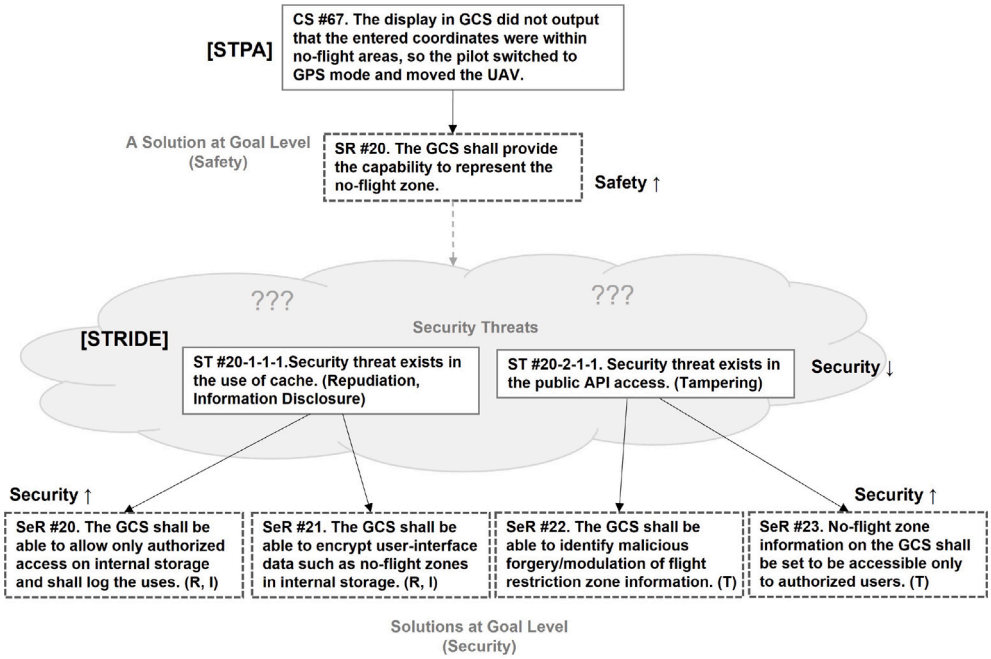

Fig. 7. A trace of co-engineering (security analysis part) without the goal-tree analysis.

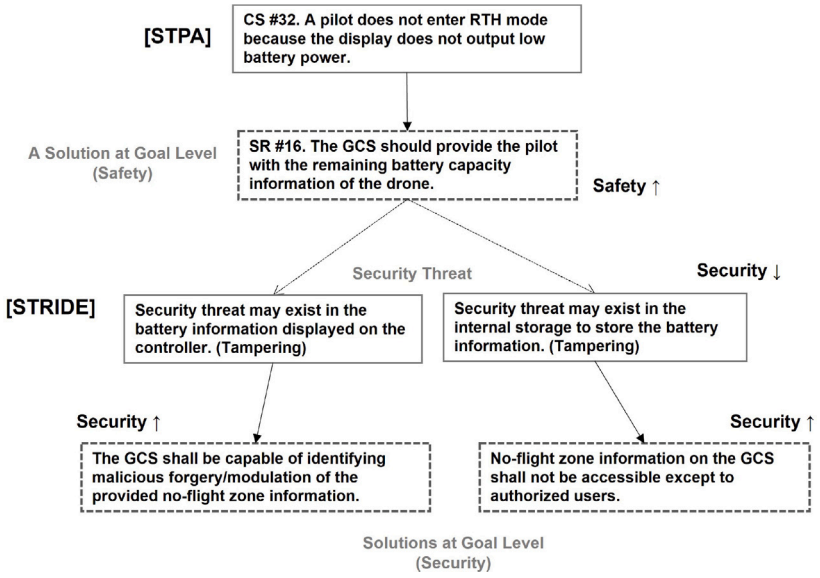

Fig. 8. A trace of co-engineering (security analysis part), which may not need the goal-tree analysis.

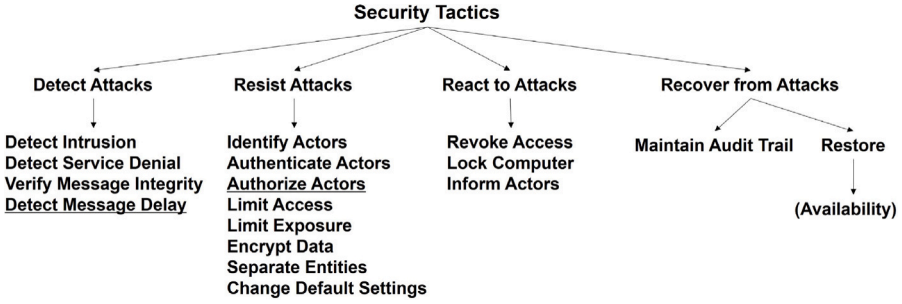

Fig. 9. Tactics for security [49].

then tries to refine the high-level security requirements into specific functional ones too, and it ends one cycle of the analysis activity in requirements engineering process. The sequence of safety analysis, goal-tree refinement, security analysis and another goal-tree refinement is seamlessly iterated in the process of requirements engineering, where any conflict of requirements will have an opportunity to be resolved. Our case study of a simplified UAV example uses STPA and STRIDE techniques for safety and security analysis respectively, and shows that the proposed approach is fully applicable up to industrial cases.

We are now focusing on how systematically identify, analyze and resolve various conflicts between safety and security requirements, which the iterative process of requirements engineering keeps causing. We are planning to apply a variant of goal-trees, which can provide useful notations to visualize the conflicts comprehensively. It seems necessary first to classify and prioritize conflicts between different types of requirements at different levels, with the support of requirement management and analysis tools. We are also considering how safety and security validation plans may affect the coordinating safety and security at the requirement validation phase.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Junbeom Yoo reports financial support was provided by National Research Foundation of Korea.

Data availability

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

Acknowledgments

This paper was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1F1A1047246).

References

[1] Recommendations for Security and Safety Co-engineering, Tech. Rep., MERgE ITEA2 - Project 11011, 2016.

[2] S. Paul, L. Rioux, Over 20 years of research into cybersecurity and safety engineering: A short bibliography, WIT Trans. Built Environ. 151 (2015) 335–349, http://dx.doi.org/10.2495/SAFE150291.

[3] IEC 62859, Nuclear power plants - Instrumentation and control systems - Requirements for coordinating safety and cybersecurit, 2016.

[4] Functional Safety of Electrical, Electronic and Programmable Electronic (E/E/PE) Safety-Related Systems (IEC 61508), Standard, International Electrotechnical Commission (IEC), 2000.

[5] Nuclear Power Plants - Instrumentation and Control Systems Important to Safety - Software Aspects for Computer-Based Systems Performing Category a Functions, Standard, International Electrotechnical Commission (IEC), 2006.

[6] Nuclear Power Plants - Instrumentation and Control Systems Important to Safety - Software Aspects for Computer-Based Systems Performing Category B or C Functions, Standard, International Electrotechnical Commission (IEC), 2018.

[7] X. Lyu, Y. Ding, S.-H. Yang, Safety and security risk assessment in cyber-physical systems, IET Cyber-Phys. Syst. Theory Appl. 4 (3) (2019) 221–232.

[8] S. Sadvandi, N. Chapon, L. Piètre-Cambacédès, Safety and security interdependencies in complex systems and sos: Challenges and perspectives, in: Complex Systems Design and Management, Springer Berlin Heidelberg, 2012, pp. 229–241.

[9] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, Y. Halgand, A survey of approaches combining safety and security for industrial control systems, Reliab. Eng. Syst. Saf. 139 (2015) 156–178, http://dx.doi.org/10.1016/j.ress.2015.02.008, URL https://www.sciencedirect.com/science/article/pii/S0951832015000538.

[10] E. Lisova, I. Šljivo, A. Čaušević, Safety and security co-analyses: A systematic literature review, IEEE Syst. J. 13 (3) (2019) 2189–2200, http://dx.doi.org/10.1109/JSYST.2018.2881017.

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

[12] L. Shan, B. Sangchoolie, P. Folkesson, J. Vinter, E. Schoitsch, C. Loiseaux, A survey on the applicability of safety, security and privacy standards in developing dependable systems, in: A. Romanovsky, E. Troubitsyna, I. Gashi, E. Schoitsch, F. Bitsch (Eds.), Computer Safety, Reliability, and Security, Springer International Publishing, Cham, 2019, pp. 74–86.

[13] B. Nuseibeh, S. Easterbrook, Requirements engineering: A roadmap, in: Proceedings of the Conference on the Future of Software Engineering, ICSE '00, 2000, pp. 35–46.

[14] IEEE Standard for Developing a Software Project Life Cycle Process, IEEE Std 1074-2006 (Revision of IEEE Std 1074-1997), 2006, pp. 1–110, http://dx.doi.org/10.1109/IEEESTD.2006.219190.

[15] IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998, 1998, pp. 1–40, http://dx.doi.org/10.1109/IEEESTD.1998.88286.

[16] A. van Lamsweerde, Goal-oriented requirements engineering: A guided tour, in: 5th IEEE International Symposium on Requirements Engineering, RE'01, 2001.

[17] V. Bolbot, G. Theotokatos, L.M. Bujorianu, E. Boulougouris, D. Vassalos, Vulnerabilities and safety assurance methods in cyber-physical systems: A comprehensive review, Reliab. Eng. Syst. Saf. 182 (2019) 179–193, http://dx.doi.org/10.1016/j.ress.2018.09.004.

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

[19] N.G. Leveson, Safeware: System Safety and Computers, Association for Computing Machinery, New York, NY, USA, 1995.

[20] C.A. Ericson, Hazard Analysis Techniques for System Safety, John Wiley and Sons, 2015.

[21] G. Kavallieratos, S. Katsikas, V. Gkioulos, Cybersecurity and safety co-engineering of cyberphysical systems - A comprehensive survey, Future Internet 12 (4) (2020) http://dx.doi.org/10.3390/fi12040065.

[22] M. Steiner, P. Liggesmeyer, Qualitative and quantitative analysis of CFTs taking security causes into account, in: Computer Safety, Reliability, and Security, 2015, pp. 109–120.

[23] G. Macher, A. Höller, H. Sporer, E. Armengaud, C. Kreiner, A combined safety-hazards and security-threat analysis method for automotive systems, in: Computer Safety, Reliability, and Security, 2015, pp. 237–250.

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

[25] S. Plósz, C. Schmittner, P. Varga, Combining safety and security analysis for industrial collaborative automation systems, in: Computer Safety, Reliability, and Security, 2017, pp. 187–198.

[26] E. Ruijters, S. Schivo, M. Stoelinga, A. Rensink, Uniform analysis of fault trees through model transformations, in: 2017 Annual Reliability and Maintainability Symposium, RAMS, 2017, pp. 1–7, http://dx.doi.org/10.1109/RAM.2017.7889759.

[27] N.G. Leveson, Engineering a Safer World: Systems Thinking Applied to Safety, MIT press Cambridge, 2011.

[28] N.G. Leveson, J.P. Thomas, STPA Handbook, Cambridge, MA, USA, 2018.

[29] W. Young, N.G. Leveson, An integrated approach to safety and security based on systems theory, Commun. ACM 57 (2) (2014) 31–35.

[30] Y. Heo, D.-A. Lee, J. Yoo, An approach to analyze security and safety requirements for UAV using STPA-Sec, in: Korea Software Congress 2020, KSC2020, 2020, pp. 183–185 (in Korean).

[31] M.L. Salgado, M.S. de Sousa, Cybersecurity in aviation: the STPA-sec method applied to the tcas security, in: 2021 10th Latin-American Symposium on Dependable Computing, LADC, IEEE, 2021, pp. 1–10.

[32] J.M. Sayers, B.E. Feighery, M.T. Span, A STPA-Sec case study: Eliciting early security requirements for a small unmanned aerial system, in: 2020 IEEE Systems Security Symposium, SSS, IEEE, 2020, pp. 1–8.

[33] E. Troubitsyna, An integrated approach to deriving safety and security requirements from safety cases, in: 2016 IEEE 40th Annual Computer Software and Applications Conference, Vol. 2, COMPSAC, IEEE, 2016, pp. 614–615.

[34] C. Ponsard, G. Dallons, P. Massonet, Goal-oriented co-engineering of security and safety requirements in cyber-physical systems, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2016, pp. 334–345.

[35] G. Sabaliauskaite, L.S. Liew, J. Cui, Integrating autonomous vehicle safety and security analysis using stpa method and the six-step model, Int. J. Adv. Secur. 11 (1 and 2) (2018) 160–169.

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

[37] I. Friedberg, K. McLaughlin, P. Smith, D. Laverty, S. Sezer, STPA-SafeSec: Safety and security analysis for cyber-physical systems, J. Inf. Secur. Appl. 34 (2017) 183–196, http://dx.doi.org/10.1016/j.jisa.2016.05.008.

[38] W.G. Temple, Y. Wu, B. Chen, Z. Kalbarczyk, Systems-theoretic likelihood and severity analysis for safety and security co-engineering, in: Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification, 2017, pp. 51–67.

[39] G. Kavallieratos, S. Katsikas, V. Gkioulos, SafeSec Tropos: Joint security and safety requirements elicitation, Comput. Stand. Interfaces 70 (2020) 103429, http://dx.doi.org/10.1016/j.csi.2020.103429.

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

[41] Y.-R. Chen, S.-J. Chen, P.-A. Hsiung, I.-H. Chou, Unified security and safety risk assessment - A case study on nuclear power plant, in: 2014 International Conference on Trustworthy Systems and their Applications, 2014, pp. 22–28, http://dx.doi.org/10.1109/TSA.2014.13.

[42] S. Aljahdali, J. Bano, N. Hundewale, Goal oriented requirements engineering - A review, in: ISCA International Conference on Computer Applications in Industry and Engineering, CAINE 2011, 2011.

[43] ISO/IEC 25010, ISO/IEC 25010:2011, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2011.

[44] A. Shostack, Threat Modeling: Designing for Security, Wiley, 2014.

[45] Microsoft, The STRIDE threat model, 2009.

[46] R. Khan, K. McLaughlin, D. Laverty, S. Sezer, STRIDE-based threat modeling for cyber-physical systems, in: 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT-Europe, 2017, pp. 1–6, http://dx.doi.org/10.1109/ISGTEurope.2017.8260283.

[47] E.B. Fernandez, Threat modeling in cyber-physical systems, in: 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, DASC/PiCom/DataCom/CyberSciTech, 2016, pp. 448–453, http://dx.doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2016.89.

[48] S. Abdullahi, M. Zayyad, N. Yusuf, L. Idris Bagiwa, A. Nura, A. Zakari, B. Dansambo, Software requirements negotiation: A review on challenges, Int. J. Innov. Comput. 11 (2021) 1–6.

[49] J. Bass, L.J. Bass, R. Kazman, Software Architecture in Practice, 3rd ed., Pearson, 2013.

[50] S. Winkler, J. von Pilgrim, A survey of traceability in requirements engineering and model-driven development, Softw. Syst. Model. 9 (4) (2010) 529–565.

[51] S. Jung, A Comprehensive Relationship Analysis for Heterogeneous Artifacts in Multiple-Collaborative Safety-Critical Systems (Ph.D. thesis), Konkuk University, 2022.

[52] T. Gu, M. Lu, L. Li, Extracting interdependent requirements and resolving conflicted requirements of safety and security for industrial control systems, in: 2015 First International Conference on Reliability Systems Engineering, ICRSE, 2015, pp. 1–8, http://dx.doi.org/10.1109/ICRSE.2015.7366481.