IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 December 2022, accepted 8 January 2023, date of publication 19 January 2023, date of current version 2 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3238326

RESEARCH ARTICLE

Evaluation of Visual Notations as a Basis for ICS Security Design Decisions

SARAH FLUCHS[1,3], RAINER DRATH[2], AND ALEXANDER FAY[3], (Senior Member, IEEE)

[1]admeritia GmbH, 40764 Langenfeld, Germany
[2]School of Engineering, Pforzheim University, 75175 Pforzheim, Germany
[3]Department of Automation, Helmut Schmidt University, 22043 Hamburg, Germany

Corresponding author: Sarah Fluchs (sarah.fluchs@admeritia.de)

This work was supported by the German Federal Ministry of Education and Research (BMBF) under Grant 16KIS1269K.

ABSTRACT For making informed security decisions during the design of industrial control systems (ICS), engineers need to process large amounts of security-relevant information outside their area of expertise. This problem moves the presentation of the security-relevant information into focus: security-relevant engineering information must be presented to security decision-makers in a way that enables them to decide upon security measures to build a defensible system. Visual representations have the potential to effectively convey suchlike information, thus saving the engineers' brain capacity for the security decision-making. However, research shows that this potential is only realized if the visualizations are carefully constructed for cognitive effectiveness. As a prerequisite for constructing a visual language for security engineering in the future, this paper explores two scientific questions: 1) what are the requirements for visualizing security-relevant engineering information in a way that enables engineers to make security decisions during ICS design? and 2) which existing visual languages meet (parts of) these requirements? The evaluation of existing visualizations reveals that there is a need for an improved, specialized visual language for security engineering that builds upon established engineering visualizations like piping and instrumentation diagrams and network maps, represents all security-relevant information as icons to achieve semantic transparency, and includes filtering mechanisms to reduce the complexity of each single diagram. The paper finishes with defining the main pillars of a future visual language that should allow ICS engineers to quickly capture security-relevant information and guide them through the process of selecting the right security measures to design a defensible ICS.

INDEX TERMS Automation engineering, industrial control system security, security by design, visual language.

I. INTRODUCTION

During the engineering of industrial control systems (ICS), many cybersecurity decisions are inevitably being made, because many of the architecture and configuration choices made during design also affect the system's cybersecurity. The sum of these decisions determines if and how the control system, once operational, will be defensible against cybersecurity threats. However, security decisions during design are often not made consciously, let alone systematically. This leads to security decisions being overlooked until it's too late to make them (at a reasonable cost) [1], [2] and consequently, insecure ICS.

Perhaps the biggest challenge in making security design decisions – to be more precise, selecting appropriate security measures – lies in distinguishing the security-relevant information from the irrelevant in a large pool of engineering information [2], [4].

Existing approaches identify relevant information about the system to be protected for the identification of vulnerabilities, attack paths, or risk. For the identification of vulnerabilities, a simple list of products used in the system under consideration suffices to assign applicable vulnerabilities from a database [7]. For the identification of attack

The associate editor coordinating the review of this manuscript and approving it for publication was Arianna Dulizia.

paths or risks, a model of the system's components and their relations is required. This information is then combined with knowledge about known vulnerabilities, software weaknesses, attack techniques, and assumptions about attackers' goals and / or skills, to identify attack paths or risks [8], [9], [10], [11], [12]. Notably, all these approaches terminate at vulnerability, risk, or attack path identification. They do not identify, let alone process, relevant information to decide upon security measures. Some of the authors even explicitly state that they regard supporting systematic identification of security measures using their approaches a challenge [4], [12].

If security-relevant information for supporting the actual decision-making is identified, measures are assumed to be selected only based on compliance with regulation. In this case, the relevant system information is reduced to basic system type information to clarify if a security measure from a specific standard or regulation is or isn't applicable to the system [13]. But in reality, decisions about secure design need to consider additional factors besides compliance: Security measures also need to be technically feasible, prevent the most critical attack scenarios, not restrict functional requirements, and fit the project's budget.

In [14], the authors have proposed four concepts representing the security-relevant information that should be extracted from engineering data to support security design decision-making: (1) unwanted events that security design needs to prevent, (2) a system's functions (including architecture and data flow) that could contribute to these unwanted events, (3) security parameters that contain design decisions potentially affecting security, and (4) attack indicators flagging critical security parameter values that may be used in a cybersecurity attack. Also, security decision-making paths are described and how each decision-making path uses the security-relevant information for identifying security measures.

From the four concepts named above, it becomes clear that the security-relevant information for decision-making has two challenging characteristics: It is numerous, and it spans across a wide range of knowledge domains. Understanding system functions and unwanted events requires system-specific engineering knowledge from different domains of the highly interdisciplinary automation engineering workflow [5], [6], e.g. the physical process, safety, I/O, PLC's, the control system, the HMI, or networking. Understanding attack indicators requires security knowledge. Finally, identifying and deciding about security parameters – and thus, the security measures – requires knowledge from both security and ICS engineering.

This results in a problem relevant in industrial practice: Security design decision-making requires understanding and processing large amounts of information [2], [4], most of which will be outside the area of expertise of any security decision maker, regardless of their profession. This problem moves the presentation of the security-relevant information into focus: Just like security operations information must be presented to security operators sitting in front of a security operations center (SOC) monitor in a way that enables them to decide how to react to a security incident, security engineering information must be presented to security engineers in a way that enables them to decide how to design a defensible system.

Visualizations are well suited for the problem at hand for two reasons: They have unique advantages in presenting complex information to non-experts because of the way human cognition works [20] and they are already being used during ICS engineering.

Therefore, as a prerequisite for constructing a visual language for security engineering in the future, this paper explores two scientific questions: *1) what are the requirements for visualizing security-relevant engineering information in a way that enables engineers to make security decisions during ICS design?* and *2) which existing visual languages meet (parts of) these requirements?*

The remainder of this paper is organized as follows: Chapter II introduces a security design decision-making workflow. In chapter III, a visual language for security design decision-making is motivated by outlining how it could support the decision-making workflow. In chapter IV, requirements are defined that the visual language needs to fulfill, including requirements for semantics in the form of a concept dictionary and requirements for the visual syntax, more precisely, for its cognitive effectiveness. In chapter V, existing visual notations are portrayed and evaluated against these requirements. In chapter VI, the key findings of the evaluation are summarized. Chapter VII turns these key findings into four basic pillars for a future visual language for ICS security design decision-making. Lastly, chapter VIII contains the conclusion and outlook.

II. BACKGROUND: SECURITY DESIGN DECISION-MAKING

To understand the motivation for a visual language for ICS security engineering, it is described how security design decisions can be made and where the challenges are.

A. SECURITY DESIGN VS. SECURITY OPERATIONS

In this work, decisions in security design are in focus, not those in security operations. The terms ''security design'' and ''security engineering'' are used interchangeably.

In security design, the main goal is build a defensible system, while in security operations, the main goal is to defend this system during operations. While security operations is about reacting dynamically to upcoming security challenges like newly found vulnerabilities or actual attacks on the system, security design is about creating the readiness to react dynamically during operations.

While differentiating between security design and security operations is worthwhile, the two have touching points. Security design and security operations decisions are often tied to the respective lifecycle phases (design and operations) but making security design decisions can also be necessary

during operations if a system is changed or re-engineered. To take this even further: Because many legacy systems were not built with cybersecurity in mind, security design decisions may affect existing systems to retrospectively improve their security posture even though they are otherwise not being re-engineered. Also, security design decisions, if made and documented systematically, are essential to inform security operations decisions: During operations, it should be traceable at any time why a specific security design decision has been made and if (and how) it may be revised if new information becomes available.

The basic steps are the same for making decisions in both security design and security operations:

1) INFORMATION COLLECTION

From a large amount of information the security-relevant pieces need to be found,

2) DECISION PREPARATION

the security-relevant information needs to be presented in a way that informs decision-making,

3) DECISION MAKING

finally, the actual decision needs to made and documented.

In security operations, there are many examples where security decision-making is well established:

For **information collection**, established concepts to identify security-relevant information are indicators of compromise, vulnerabilities, signatures or rules to identify known malware, or anomalies compared to a baseline.

For **decision preparation**, available tools to process this information and prepare for decision-making are antivirus tools, security monitoring and detection tools, intrusion detection and prevention tools.

For **decision-making** there are established organizational structures for making operations decisions. Simple decisions like quarantining and removing malware are made by the systems' users or even automatically. For more complex cases, such as the decision to evaluate and react to a security incident, security operations centers (SOCs) are formed, where SOC operators look at the information provided by tools and make the necessary decisions.

B. SECURITY DESIGN DECISION-MAKING

For security design, none of the three basic steps for decision-making is as well established as for operations.

Information collection: Because in security design, there is not yet an operational system, security-relevant information needs to be drawn from engineering information. In [14], the authors have first identified four types of security-relevant information that can be found in engineering information: unwanted event indicators, system functions, security parameters, and attack indicators.

Unwanted event indicators mark information that provides hints to **unwanted events** – events that should be prevented by the security engineering efforts. Suchlike events are mostly unwanted because they result in high consequences for the organization, which is why they are sometimes called high consequence events [71]. Examples are the explosion of a chemical reactor, injury of humans colliding with a robot arm, or a coal excavator falling over. Examples for unwanted event indicators are alarms, redundancies, safety functions, explosion protection requirements, operating limits of machinery or tight set points or response times.

The system's **functions** that, if misused, could contribute to the unwanted events. Functions represent the purposes the system has been designed for, including the required components, data flow, and human interactions. If misused, the system's functions can contribute to the unwanted events.

Security parameters mark information that, when changed, can influence the security of the system under consideration – or to be more precise, the system's defensibility and / or its readiness to react to security incidents. Some security parameters are obviously in the security domain, e.g. authentication or encryption mechanisms or integrity protection mechanisms for PLC logic. Others are in the control engineering domain and may not be identified as security-relevant at first sight, e.g. the mechanism to change the PLC's operating modes, the decision whether PLC logic updates during operations are allowed, or the construction (mechanical? electronic?) of a safety shutdown.

Attack indicators are security parameters values that can potentially be used in an attack. As an example, for the integrity protection of PLC logic parameter, no integrity protection or integrity checking through a checksum yield the potential to manipulate PLC logic without anyone noticing.

Decision preparation and decision-making: Because the concepts for relevant information for security design are not well-established yet, neither are the methods to process them. Consequently, there are also no ''security design tools'' that are as ubiquitous as the equivalent security operations tools like antivirus, security monitoring or intrusion detection.

Therefore, at this point, security decision-making in design based on the security-relevant design information identified above can only be described as a workflow. In [14], the authors have analyzed potential security design decision-making workflows from software engineering, systems engineering, and requirements engineering and identified three security design decision-making paths: goal-driven, risk-driven and compliance-driven. These three paths can be condensed into one security engineering decision-making workflow if some variants are provided within the workflow steps. The workflow steps, along with the guiding questions for each step, are summarized in Table 1.

In Table 1, the concepts that are needed for the security design decision-making workflow are in bold. They will be defined in more detail later.

C. ICS SECURITY DECISION-MAKING CHALLENGES

Regardless of who is charged with making security design decisions (a person called security engineer in this paper),

TABLE 1. Security design decision-making workflow and guiding questions.

ID	Security Decision-Making Workflow	Guiding questions
1	Identify **unwanted events**	Why is security engineering needed?
2	Identify the essential **entities, relations** and **functions** of the system under consideration	What parts of the system under consideration can contribute to unwanted events?
3	Identify variable **security parameters**	What options are there to influence the security of the system under consideration?
4	Identify **attack indicators** and **attack scenarios** to determine how system functions and security parameters can contribute to unwanted events	How can the system under consideration contribute to unwanted events?
5	Identify **security goals** to prevent unwanted events	What needs to be achieved in order to prevent unwanted events?
6	Set security parameters' values to meet security goals	How can the system be designed to meet security goals?

they will have to digest large amounts of information - for which they are no experts - in limited time.

It must be noted that these are three separate challenges:

(1) The sheer amount of information that a decision-maker has to consider as potentially security-relevant,

(2) the fact that some of this information will inevitably be outside their area of expertise and

(3) all this needs to be done in a limited amount of time.

While the reasons for the third challenge are obvious, the first two call for further explanation.

The first challenge – the large number of potentially security-relevant information – is due to the fact that security attacks could stem from any part of the system, no matter how unassuming or even irrelevant to the designing engineers it may be (''a system is only as secure as its weakest link''). As described in [14] and [15], security parameters are often seemingly small configuration details that do not look security-relevant at first sight. The security parameters are a long list of hundreds of parameters, each with a handful of possible values – and that is not considering variants for different ICS manufacturers. What's more, the security-relevant information is likely to change over time, as new threats or vulnerabilities emerge.

The second challenge is due to the fact that ICS security decision-making requires knowledge in at least two domains: the domain of the ICS to be secured (plant and process, ICS entities, relations, and functions, as well as unwanted events) and the domain of security (security goals, attack indicators, attack scenarios etc.). To make it worse, decision-making requires combining knowledge from both domains: to decide upon a security parameter's values, both the security implications and the potential impact on the system under consideration need to be taken into account.

This alone would suffice to argue that regardless if the security design decision-maker is an expert in the ICS or in the security domain, they will lack some required knowledge.

But the challenge is even greater: In the ICS knowledge domain, it would be a mistake to assume every control engineers is an expert for all required information, because the engineering of a plant and its automation systems is highly interdisciplinary [5], [6]. Thus, understanding the system to be secured means understanding the characteristics of a system that, for a large part, (many) others have designed. Also, security decision-making calls for gaining an overview of the system-as-a-whole that none of the involved disciplines has been needing to gain.

III. MOTIVATION FOR A VISUAL LANGUAGE FOR ICS SECURITY DECISION-MAKING

In this chapter, it is investigated if and how visualizations can address the identified challenges in security design decision-making (digesting large amounts of information outside one's area of expertise in limited time). Also, the terminology to analyze visual languages is introduced.

A. HOW VISUALIZATION CAN SUPPORT SECURITY DESIGN DECISION-MAKING

There is evidence that diagrams, containing graphical notations, have a big advantage in scenarios where non-experts need to quickly process information [16], [17], [18], [19] – which is an exact description of the challenges in security-decision-making identified earlier. This advantage is due to the way the human brain processes diagrams: visual perception takes place before cognition [20]. Also, these perceptual processes are sub-conscious (''pre-attentive'') and much faster than the conscious cognitive processes [20], [21]. Consequently, whenever a human can consume information via diagrams, the information is not only digested faster, but also cognitive resources are freed up for other tasks [20], [22].

More specifically, visualizations can help in each step of the security design decision making workflow introduced earlier – by helping to **understand** required input as well as to **decide** and **document decisions**. In the following, the benefit of visualizations in the workflow is analyzed step by step:

1) **Identify unwanted events:**
Understand: Unwanted events in ICS systems tend to originate from the process, so plant and process information is relevant input to digest when identifying

unwanted events. Plant and process information is often represented visually in the form of technical drawings like piping and instrumentation diagrams (P&IDs). Control engineers use P&IDs to identify where control loops need to be placed and for what reason.
Decide: Security engineers may use P&IDs to identify where unwanted events may happen.
Document: To document their findings, a representation of unwanted event indicators and unwanted events can be put on top of a technical drawing.

2) **Identify the essential entities, relations and functions of the system under consideration:**
Understand: This step basically encompasses understanding the system under consideration's structure and use, the system under consideration being selected from the ICS and the surrounding IT infrastructure. The required information is not security-specific except that the level of detail needed for security engineering is lower than for the design of the system: security engineers only need to understand how the system works, while system designers need to build the system so it will work. The most important goal for the security engineer is to have a good overview of the entire system (since the system is ''only as secure as its weakest link'') while the most important goal for the system designer is to display all relevant details for building the system. Therefore, network drawings and data flow drawings which are used to design the ICS serve as a good basis for system understanding to security engineers, but may need more abstraction. Also, a system's intended use, including the humans who use it, is important information that security engineers need to understand [70].
Decide: Based on their system understanding, security engineers need to decide which parts of the system could contribute to an unwanted event, or more generally, what is in scope for security engineering.
Document: To document their findings, elements in scope may be highlighted or those out of scope eliminated.

3) **Identify variable security parameters:**
Understand: It must be understood which parts and configurations of the system under consideration can influence the system's security – these are the security parameters. Because security parameters are dependent on the system that was chosen to be in scope, they are merely an additional piece of information to add to the visualization used for system understanding. This also has the advantage of seeing how the system is structured and what contributes to its security at the same time. Thus, like for the unwanted event indicators, a representation of security parameters that can be put on top of a technical drawing, in this case a network map or data flow diagram, would be helpful.
Decide: Which of the security parameters can actually be considered as variable and which are fixed because of non-security requirements.
Document: Security parameters must look different if variable or fixed.

4) **Identify attack indicators and attack scenarios to determine how system functions and security parameters can contribute to unwanted events:**
Understand: To understand how the system under consideration (or parts of it) can contribute to unwanted events, it is interesting to know which system entities provide opportunities for attack (attack indicators).
Decide & document: Seeing system structure and attack indicators at the same time is a good basis for identifying attack scenarios, which are pathways through the system using attack indicators as stepping stones. For this reason, visualizations are already common for attack paths.

5) **Identify security goals to prevent unwanted events** and

6) **Set security parameters' values to meet security goals:**
Understand: This is where the information from the previous workflow comes together: The unwanted events from the plant or process, the system under consideration with its functions and variable security parameters, the attack indicators and attack scenarios are all relevant information that needs to be digested to make the final security design decisions. Therefore, ideally, they can all be displayed at the same time.
Decide & document: This final security design decisions consists of defining security goals to prevent the unwanted events, and then setting a set of security parameters to specific values to achieve a specific security goal.

From the step-by-step analysis, a pattern can be observed: Each decision can only be made through the understanding of a context for which a visualization already exists, and then changing something in this context.

The understanding mostly focuses on overviewing the architecture of a (technical) system: plant, process, control system network, etc., which is common to be represented visually: the technical drawing or P&ID for step 1 (identifying unwanted events), abstract network maps and data flow diagrams for steps 2 to 6, and additionally attack graph diagrams for step 4 (identifying attack scenarios).

The decisions and documentation are then manifested in picking, highlighting, or changing certain aspects within this technical system.

The prevalence of diagrams in technical design is of course no coincidence. The defining characteristic of diagrams is that they represent information spatially, setting represented elements into relation [20]. This makes them well-suited to document complex, interconnected systems – like process plants or IT networks.

To summarize, there are two motivations to use visualization in security design decision-making. First, the decision-making requires humans to quickly digest large amounts of information for which there are no experts – a situation for which visualizations have proven advantages due to how human cognition works. Second, many (non-security) design decisions are being made using visualizations anyway, so it is easier for engineers to build upon these for security engineering as well.

B. COGNITIVE EFFECTIVENESS

The authors' primary reason for using visualizations for security decision-making is to make information easier to process for non-expert, human decision-makers.

But although it is often assumed (there is even the popular adage ''a picture says more than thousand words''), not all visualizations make information easier to comprehend. As emphasized by Larkin and Simon [18] and later by Moody [20], this is only the case if they are carefully designed for that purpose. There is a term for the ''speed, ease, and accuracy with which a representation can be processed by the human mind'' [20]: **cognitive effectiveness**. Therefore, the foundational requirement is to design visualizations for all security concepts that achieve a high cognitive effectiveness.

C. VISUAL LANGUAGE TERMINOLOGY

Moving forward, basic terms regarding (visual) languages need to be defined.

For any language, it's important to differentiate between abstract syntax, concrete syntax, and semantics. For this paper, the definitions provided in [23] are being used: The **abstract syntax** defines the constructs or notation that a language is made up of, and the rules to use it. For a natural language, this would be types of words (nouns, verbs, adjectives…) and the grammar rules specifying how to build a correct sentence. The **concrete syntax** defines how the language's constructs are expressed. For a written natural language, this could be letters of the latin alphabet, but also braille symbols, as well as compositional rules to build words and sentences from letters. The **semantics** define what each construct in a language means. For a natural language, they make sure everyone understands what a person refers to when they for example say ''house'' or ''ball''. Visual language defines a concrete (visual) syntax.

A visual language according to [20] consists of a **visual vocabulary** (graphical **symbols** representing the constructs) and a **visual grammar** (compositional rules for these symbols) that form the concrete syntax. A **visual sentence** or **diagram** is a valid expression using the concrete syntax. **Visual semantics** define the meaning of each graphical symbol by mapping them to the construct they represent. Ulrich Frank [24] suggests to build a **concept dictionary** as a definition of all constructs to be covered by a visual language. This recommendation is followed in the next chapter of this paper.

IV. REQUIREMENT ANALYSIS

In the following, the authors formulate requirements for a visual language for ICS security decision-making. On the top level, there are only two requirements:

R1 Semantics: Coverage of security engineering concepts. The language should be able to accompany the entire security design decision-making workflow introduced in chapter II.

R2 Visual syntax: Cognitive effectiveness for human non-security experts. The visual vocabulary and grammar should facilitate humans (automation engineers, non-security experts) to quickly understand the concepts from the dictionary.

A. SEMANTICS: CONCEPT DICTIONARY

To concretize requirement R1, the concept dictionary is used that was derived in [14]. Table 2 provides a detailed description of and examples for all eight concepts that are used in the security design decision-making workflow introduced in chapters II and III.

Consequently, requirement R1 may be detailed into eight subordinate **requirements R1.1-R1.8**. These requirements demand the visual security decision-making language to represent all eight concepts from the concept dictionary [14]: **entity and relation, function, security parameter, security goal, unwanted event indicator, unwanted event, attack indicator, and attack scenario**.

B. VISUAL SYNTAX: COGNITIVE EFFECTIVENESS

Requirement R2 needs further specification as well. In his seminal work ''The Physics of Notations'' [20] and its predecessor ''What Makes a Good Diagram?'' [22], Daniel Moody defines criteria for constructing diagrams with a high cognitive effectiveness. A variation of these criteria is proposed by Frank in [24]. A selection from these criteria is used as the **requirements R2.1-R2.7 for cognitive effectiveness**. To make the following chapters (which will reference the requirements that will be stated now) easier to read, they are described as concrete as possible and without the technical terms for the cognitive effectiveness principles introduced by Moody. However, Moody's more precise technical terms are added in brackets in each requirement title.

R2.1 One symbol per concept, one concept per symbol (semiotic clarity): For each concept in the concept dictionary, there should be exactly one graphical symbol. The graphical symbol should only be used for one concept, and each concept should only be represented by one symbol. These requirements are summarized as the ''principle of semiotic clarity'' in [20].

R2.2 Limited (<10) number of different symbols (graphic economy): If the reader must learn too many different symbols, diagrams become hard to read. This is because in that case, the viewer begins to consciously think about the diagram (''what did this shape mean again?''), which leads to the visualization losing its advantage of being perceived with

TABLE 2. Semantic requirements: concepts that need to be represented visually to support security design decision-making [14].

ID	Concept	Description	Examples
R1.1	**Entity and relation**	Elements that define the scope of the system to be protected.	hardware layout, network maps
R1.2	**Function**	The action for which a thing exists or is employed. It is what a system does; it is the activities, operations, and transformations that cause, create, or contribute to performance [45]. Represents the functional requirements for the systems to be secured.	"Verb + Object", e.g. open the door, program a PLC, backup PLC logic
R1.3	**Security parameter**	Representation of a security decision within the engineering workflow, highlighting a design choice that influences the security of the system to be protected.	mechanism for changing a PLC's operating modes:{key switch; password; none}, vulnerability CVE-2035-12345: {patched, unpatched}
R1.4	**Security goal**	Represents the purpose that is to be achieved by making a security decision.	integrity of output values for pump X and heater Y
R1.5	**Unwanted event indicator**	Information from the ICS engineering workflow indicating where unwanted consequences for the system to be protected could ensue.	an alarm threshold, a redundancy, an operating limit
R1.6	**Unwanted event**	Unwanted state of the system to be protected.	overpressure in reactor
R1.7	**Attack indicator**	Information from the ICS engineering workflow that indicates possible points of attack that could be used in an attack scenario targeting a function, causing an unwanted event.	mechanism for changing a PLC's operating modes: password, vulnerability CVE-2035-12345: unpatched
R1.8	**Attack scenario**	Sequence of events exploiting one or multiple entities' attack indicators to eventually cause an unwanted event. Can be modeled as a sequence of attack techniques, e.g. by [66].	Step 1: Spearphishing attachment, Step 2: User Execution, Step 3: Modify Control Logic

fast, efficient, sub-conscious mechanisms. A rule of thumb for the number of different graphics that is manageable is seven plus / minus two. The reason is believed to be limits in human working memory capacity, which is limited to about seven concepts at a time [22].

R2.3 Use of unique icons plus additional visual variables for discrimination (perceptual discriminability, visual expressiveness, semantic transparency): Different symbols should be clearly distinguishable from each other (''principle of perceptual discriminability''). They become better distinguishable if they differ as many visual variables as possible (''principle of visual expressiveness''). Visual variables, as defined in [16] and [20] are shape, color, brightness, texture, size, orientation and the horizontal and vertical position. According to [24], however, shape is the most effective differentiator, and both [20] and [24] describe the outstanding effectiveness of icons. Icons are symbols which do not need to be ''learned'', since their meaning is intuitively conveyed through their shape (''principle of semantic transparency''). On top of that, according to [20], visual elements can be recognized sub-consciously if they have unique values for at least one visual variable. Hence, not only the use of icons, but the use of unique icons per concept is required.

R2.4 Use of text for additional information or as an interpretation reminder, but not for discrimination (dual coding): Text should not be used as the only means to distinguish between concepts. However, it can be effective for providing additional information that is not well suited to be represented graphically. Also, text can serve as an interpretation reminder if the concepts themselves are lacking semantic transparency, i.e. are not completely intuitive on their own (dual coding) [20].

R2.5 Collision avoidance with common visual concepts from automation engineering (cognitive fit): The principle of cognitive fit says that different diagram dialects may be necessary for different target groups [20]. The target group of the security decision-making diagrams, automation engineers, is used to working with diagrams in their engineering workflow. This implies both chances and risks. If a concept is popular among the target group, they are likely to recognize similar concepts more easily. On the other hand, visual notations too similar to existing ones are at risk for being confused and thus misinterpreted.

R2.6 Modularized diagrams with limited number of elements (complexity management): Too much information in a diagram causes cognitive overload, making the diagram hard to comprehend. Again, working memory capacity sets limits at about seven elements to process at a time [22]. Obviously, most problems consist of more than seven elements that would need graphical representation, so diagrams need to be broken down into modules to stay comprehensible.

R2.7 Cognitive integration mechanism between diagrams: The requirement for modularized diagrams obviously leads to multiple diagrams from which the viewer needs to integrate information to comprehend the full problem [20], [22]. This calls for a mechanism to help the viewer to quickly ''anchor'' a new diagram's content in the information they've already taken in from other diagrams.

There is a common denominator in all the cognitive effectiveness requirements defined above. In his popular book for intuitive web design [25], Steve Krug introduces the principle ''don't make me think''. This means that intuitively designed web applications can be navigated by users without thinking about how to navigate them, letting users focus all their attention on the task at hand. In a similar fashion, good security engineering diagrams should not *make* engineers think (about the diagrams) either. Instead, they should *help* engineers think clearly to make security decisions. They must clarify, not complicate security decision-making. If the viewer needs to learn a visual vocabulary and think hard about what he's

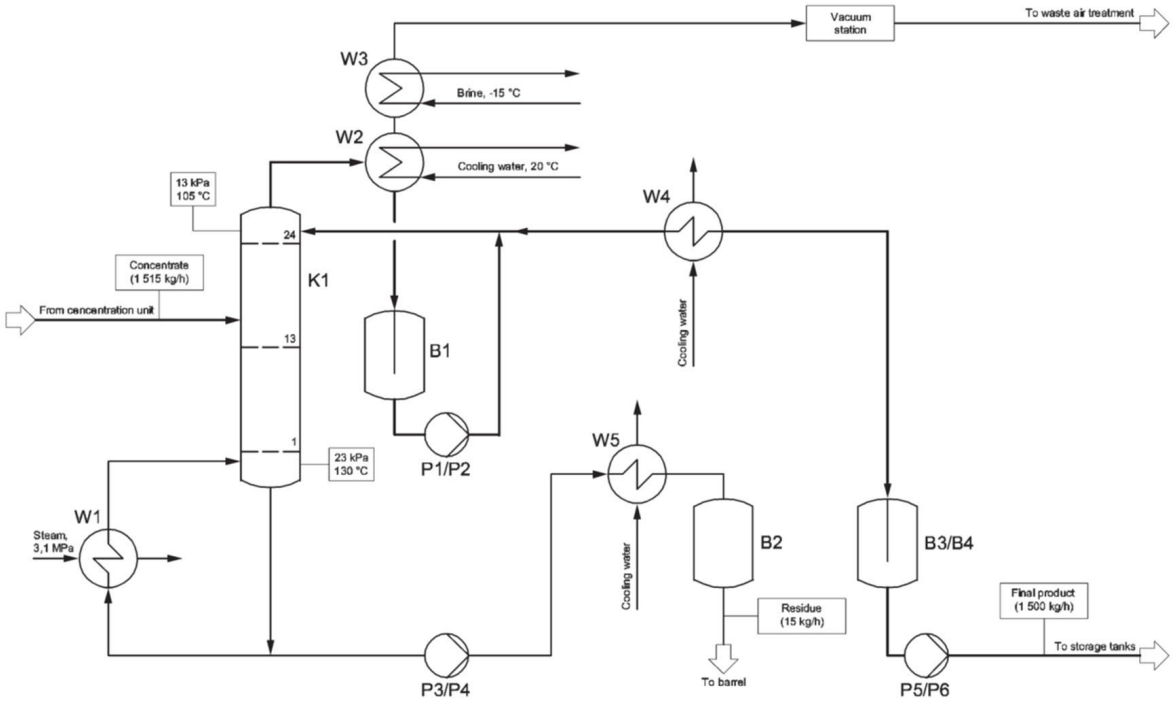


FIGURE 1. Example for a P&I diagram from ISO 10628-1 [27].

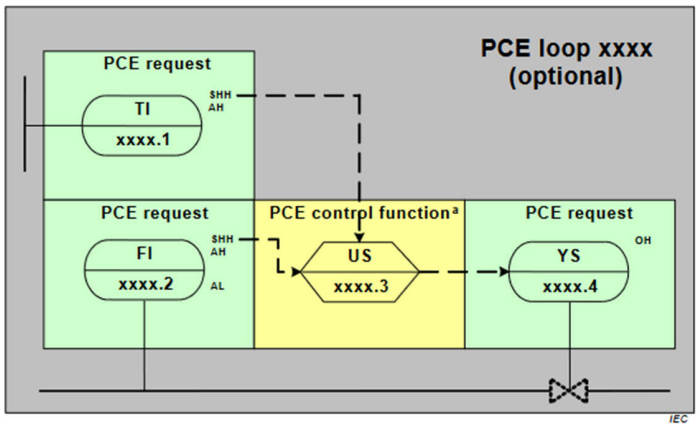


FIGURE 2. Visual representation of PCE requests in a P&I diagram according to IEC 62424:2016 [29].

seeing while looking at a diagram, it does not reduce but increase cognitive workload. To summarize, a test question for a good diagram is "does it *help* me think (and not *make* me think)"?

C. COMMON VISUAL CONCEPTS FROM PROCESS AUTOMATION ENGINEERING

Notations from process automation engineering are relevant to evaluate requirement **R2.5** (collision avoidance with common visual concepts from automation engineering). This section gives a brief overview over the most popular notations found in practical engineering projects. This is based on two sources: Research of automation engineering workflows in literature [5], [6], [26] and the review of automation engineering workflows at a chemicals producer (INEOS) and a component manufacturer (HIMA). The goal of this section is not to explain or evaluate the notations, but to be aware of them so it can be explained where they collide with the security notations introduced in chapter V.

One of the most popular diagrams in process automation engineering projects is the **piping & instrumentation diagram (P&ID)**. It consists of visual representations of the process equipment (containers, pipes, pumps, valves, motors, heaters…). P&ID diagrams and symbols are different for different industries. For the chemical and petrochemical industry, they are standardized in ISO 10628-1:2014 (diagrams) [27] and 10628-2:2012 (graphical symbols) [28]. An example P&ID is shown in Fig. 1.

Automation engineers add information about sensors, actuators, and control loops using a notation called **process**

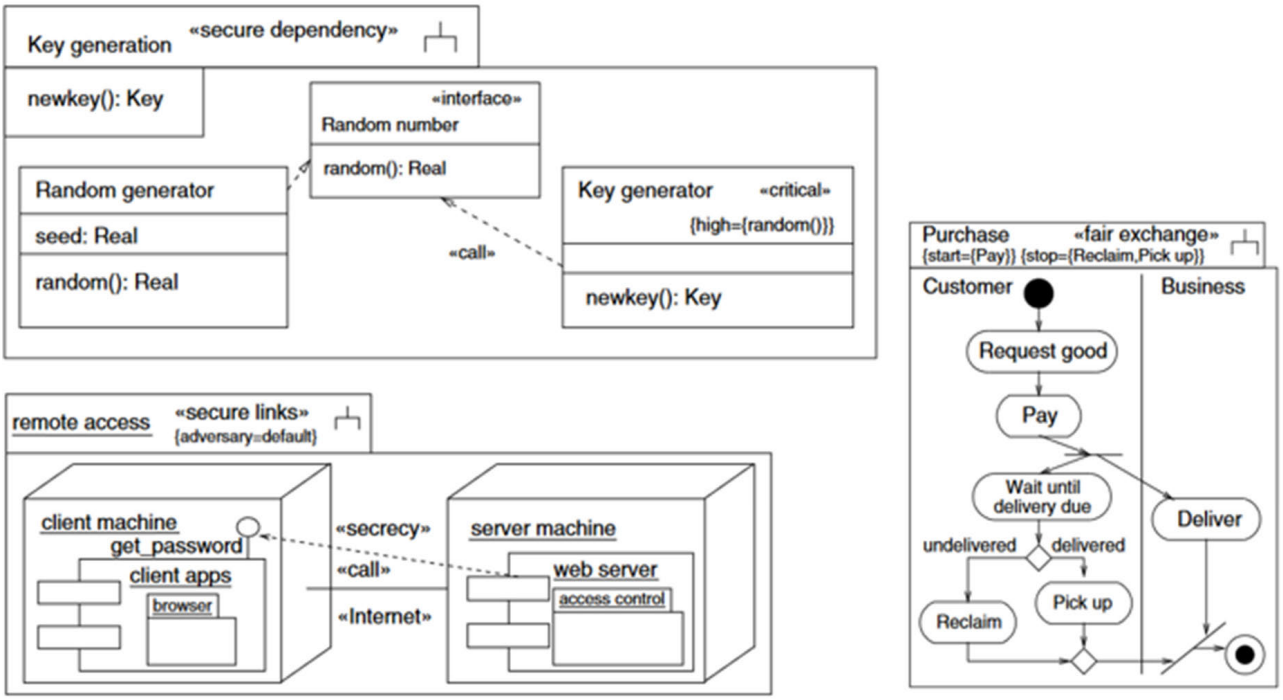


FIGURE 3. UML class diagram (top left), deployment diagram (bottom left) and activity diagram (right) including UMLsec stereotypes ≪secure dependency≫, ≪secure links≫, and ≪fair exchange≫ [36].

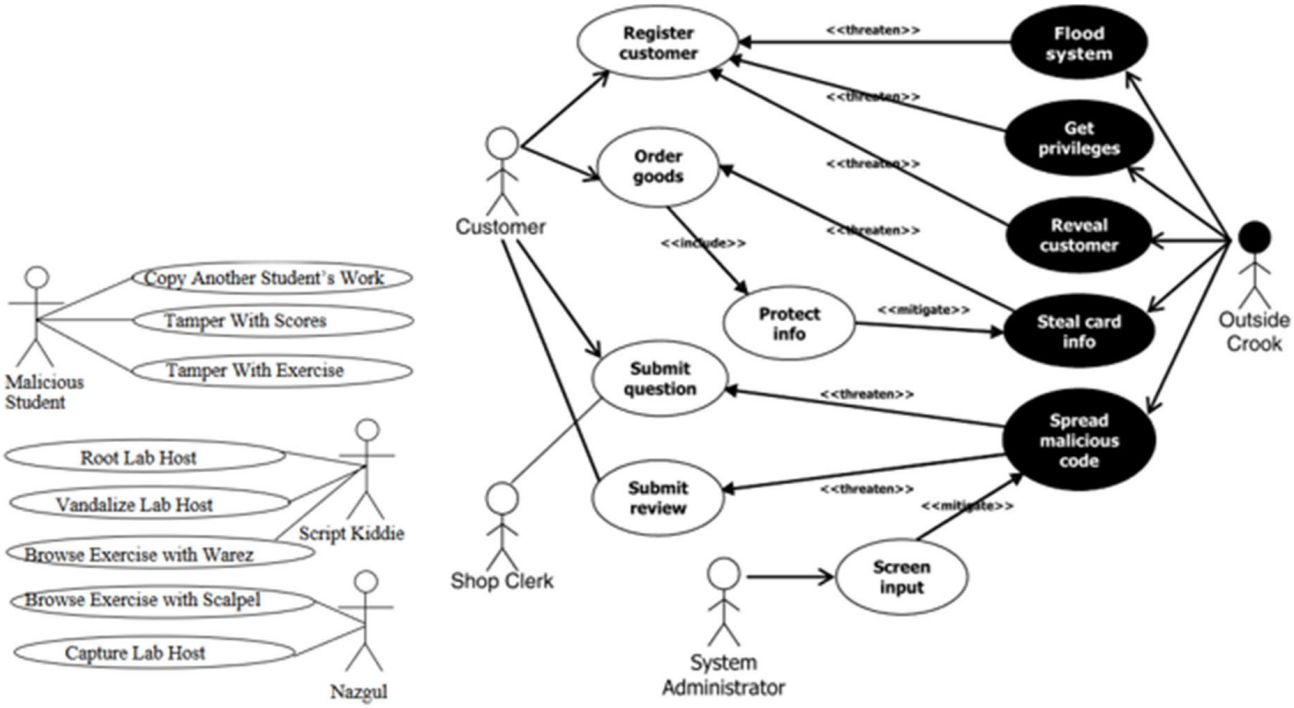


FIGURE 4. Abuse case / misuse case diagrams introduced by McDermott and Fox [38] (left) and Sindre and Opdahl [39] (right).

control engineering (PCE) requests, standardized in IEC 62424:2016 [29]. PCE requests are represented by ellipses for sensors and actuators and hexagons for control functions, which contain standardized letter codes to indicate details about the nature of the measured or controlled values (Fig. 2): What is measured (temperature, flow…)? How is it measured (in discrete or continuous values)? Are there thresholds for alarms or shutdowns? Which measurements and actuators form a control function?

Apart from P&IDs, visual notations are common for programming PLC or control system logic. There are five programming languages for PLCs defined in IEC

TABLE 3. Cognitive effectiveness of UML-based visual languages.

ID	Requirement	Met?	Comment
R2.1	**One symbol** per concept, one concept per symbol		In different diagram types, the same concept may be represented by different symbols.
R2.2	**Limited (<10) number** of different symbols		Especially relations have a large variety of different symbols.
R2.3	Use of **unique icons** plus additional visual variables for discrimination		No use of icons except for the stick figure representing an actor. Instead, mainly shapes (rectangles, ellipses) are used for visual discrimination. As an additional variable, sometimes color is used (e.g. black background in use case shapes for abuse cases).
R2.4	Use of **text** for additional information or as an interpretation reminder, but not for discrimination		Stereotypes that are often used for security extensions are mostly indicated via textual labels (e.g. "secure") as the only means of discrimination.
R2.5	**Collision avoidance** with common visual concepts from automation engineering	(x)	Ellipses, that are fundamental in P&I diagrams for representing PCE requests [29], are also an essential UML shape.
R2.6	**Modularized diagrams** with limited number of elements	(x)	UML provides the flexibility to modularize diagrams. But in reviewed applications, this is rarely used and instead, very complex diagrams are being created.
R2.7	**Cognitive integration** mechanism between diagrams		There is no support for integrating information from the different diagram types.

61131-3:2013 [30]. Three of those are visual: Ladder diagrams (LD), function block diagrams (FBD), and sequential function charts (SFC). The visual notation is not standardized in IEC 61131-3, but for all three languages, it usually consists of shapes, mostly rectangles and connecting lines, with no color or texture. These visual notations are so unspecific that they are not at risk for being mixed up with newly defined security engineering diagrams.

V. EVALUATION OF EXISTING VISUAL NOTATIONS FOR SECURITY DECISION-MAKING

In this section, the visual languages that have been proposed for security decision-making so far are introduced. Also, it is evaluated to what extent these languages meet the cognitive effectiveness requirements defined above (R2.x). The goal for the visual language is to build upon visual notations that are established and effective, while avoiding the mistakes that may be designed into some of them.

Covered are only visual languages that are meant to be interpreted by humans for making security decisions directly based on these diagrams. There are some approaches that use e.g. UML class diagrams to model security concepts, but for other purposes: **a)** for documenting security requirements that need to be built into a software or **b)** for automatically transforming modeled requirements into software (in model-driven software engineering) or **c)** to feed security-relevant information into a software tool in order to make security decisions (e.g. CySeMoL [31]).

Also, diagrams are not discussed here if the visualization is only a small portion within an otherwise formalized, textual environment, such as the security problem frame concept [32], [33].

A. UML SECURITY EXTENSIONS

UML is the most popular visual language in object oriented software engineering. It is published and maintained by the Object Management Group (OMG) [34]. UML consists of several diagram types which can be understood as different views on a system. The variety in diagrams means there is often more than one possibility to represent a concept, e.g. interfaces or package relationships [20].

Because UML is so popular in software engineering, there have been multiple proposals for UML extensions to represent security concepts as well. Most prominently, **Jürjens** has developed UMLsec [35], [36], which adds security-relevant information like security properties and security requirements to be used in various UML diagram types (Fig. 3). In the concept dictionary, the added information is mainly on the level of security requirements, hence security parameters set to certain values or new introduced security functions. UMLsec defines the security-relevant information as UML stereotypes like ≪secrecy≫, ≪integrity≫, ≪secure links≫ or ≪fair exchange ≫. In some cases, stereotypes pose additional requirements (UML constraints) to the diagrams for them to remain valid. For example, for ≪secure links ≫, the secrecy of all communication must be ensured. Constraints like these can only be implemented effectively with tool support. The fact that UML does include tool support provides indications that the visual language and diagrams in UML were not primarily designed to be read by humans. Also, although UML provides the possibility to define new visual notations for stereotypes, the stereotypes are only added as textual labels, enclosed by guillemets (≪), to the respective diagrams. No new visual notations are introduced. This is the second indication for UMLsec not primarily being designed to be read by humans, but to be used by tools.

UMLsec is not to be confused with the UML extension called SecureUML [37], which has a very narrow scope and only adds concepts related to role-based access control (RBAC): permission, resource, role, user, and authorization constraint. Like for UMLsec, UML stereotypes are defined, but no new visual notations.

Besides the general UML extensions for all diagrams, there are multiple approaches that use the behavioral UML

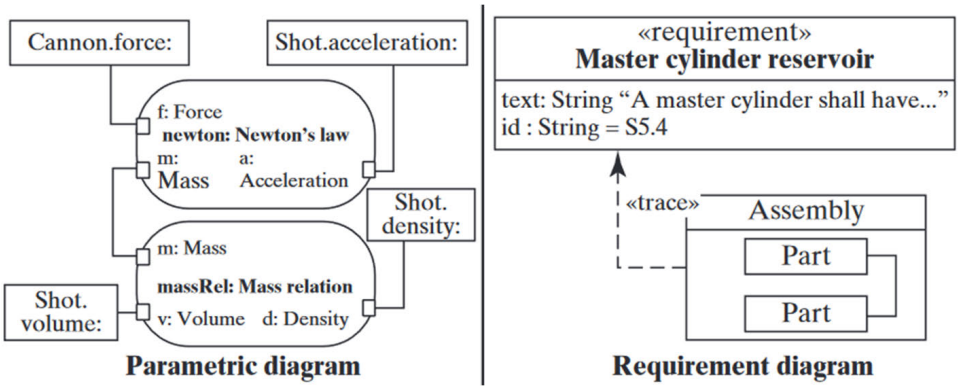


FIGURE 5. Diagram types added by SysML: parametric diagram, requirement diagram [45].

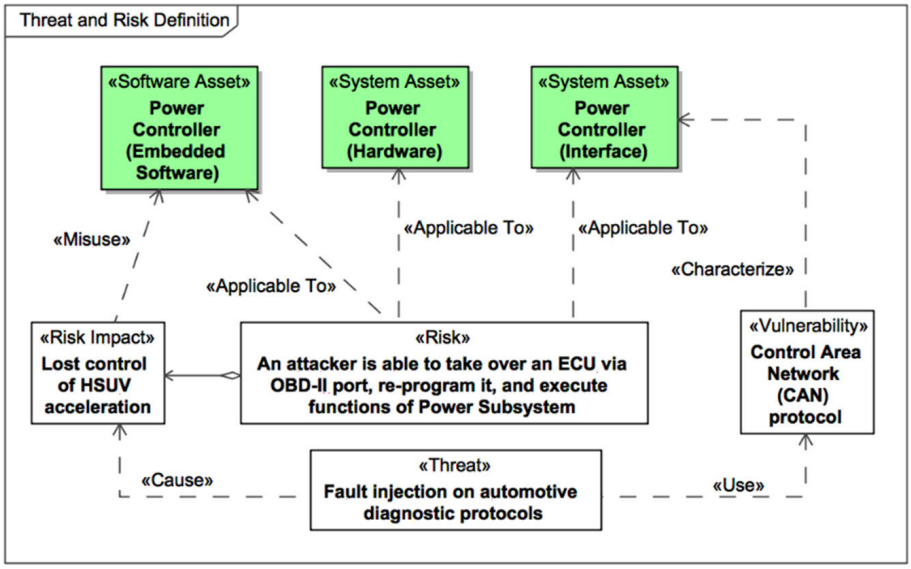


FIGURE 6. MBSEsec threat and risk definition diagram as an extension of an UML class diagram [47].

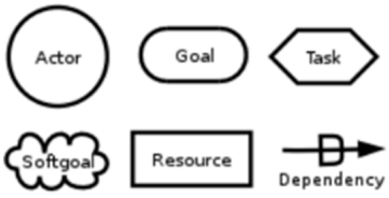


FIGURE 7. Visual notations for the fundamental concepts of i* / Tropos [49].

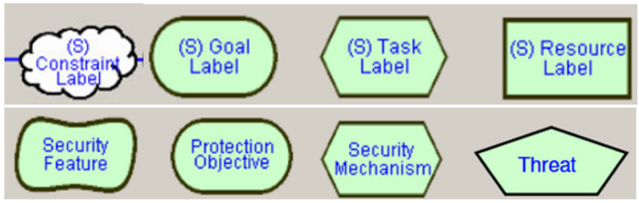


FIGURE 8. Visual notations for security-specific concepts in Secure Tropos [50].

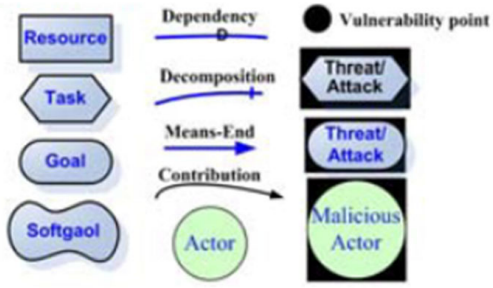


FIGURE 9. Malicious versions of the actor, goal and task concepts distinguished by their black background [55].

diagrams (use case and sequence diagrams) for security engineering purposes.

McDermott and Fox first introduce the idea of abuse cases that model malicious behavior (the equivalent in the concept dictionary being attack scenarios) [38]. Just like use cases are an easy way for non-technical people to elicit functional requirements, abuse cases are suggested to be an easy way for non-security professionals to elicit security requirements. McDermott and Fox use the normal UML use case diagrams for their abuse cases, only labeling them as malicious through annotations (e.g. "malicious actor", see Fig. 4).

Sindre and Opdahl use the same concept, but call it "misuse cases" in [39]. They visually distinguish misuse

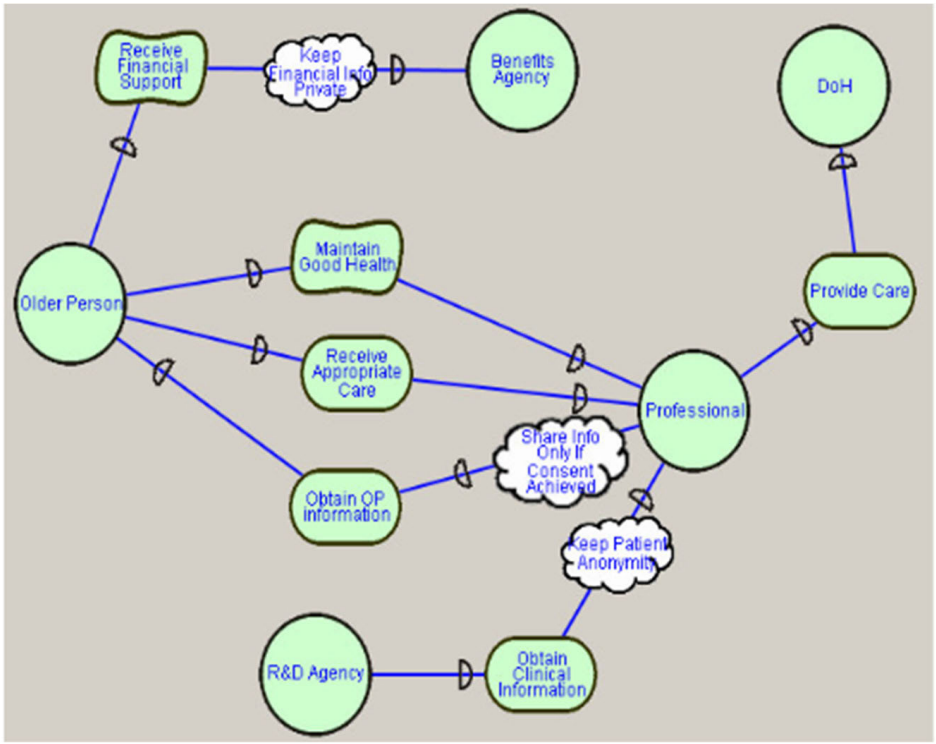


FIGURE 10. Example for an actor diagram in Secure Tropos, containing actors (circles), dependencies (relationsl, (soft) goal (ellipses), and security constraints (clouds) [50].

TABLE 4. Cognitive effectiveness of i*/Tropos-based visual languages.

ID	Requirement	Met?	Comment
R2.1	**One symbol** per concept, one concept per symbol	(x)	Generally yes, but sometimes related, but not identical concepts are represented by the same symbol (e.g. goal and protection objective).
R2.2	**Limited (<10) number** of different symbols	(x)	In the basic i*/Tropos language, less than 10 symbols are used. In extensions, however, this number is sometimes exceeded.
R2.3	Use of **unique icons** plus additional visual variables for discrimination		Only shapes used for visual discrimination. Color is used very heterogeneously in different i*/Tropos adaptations.
R2.4	Use of **text** for additional information or as an interpretation reminder, but not for discrimination	(x)	Generally yes, text only used to discriminate among different kinds of dependencies in Secure Tropos by Massacci and Zannone [54].
R2.5	**Collision avoidance** with common visual concepts from automation engineering		Ellipses and hexagons, which are also common in PCE requests in P&I diagrams [29], are extensively used.
R2.6	**Modularized diagrams** with limited number of elements		Diagrams for realistic scenarios become very large and complex.
R2.7	**Cognitive integration** mechanism between diagrams		In adaptations where there are multiple diagrams (e.g. actor diagrams and security reference diagrams in Secure Tropos [50]), no integration mechanism is provided.

cases from normal use cases by way of color: They have black backgrounds in all shapes, where normal use cases are white (Fig. 4).

Raspotnig et al. suggest a methodology called CHASSIS [72] (Combined Harm Assessment of Safety and Security for Information Systems). CHASSIS applies the misuse case concept introduced by Sindre and Opdahl to elicit security and safety requirements in the same workflow. The misuse cases and visualization do not differ from Sindre and Opdahl, so no new insights can be gained from the concept regarding visual notations.

Popp et al. propose security extensions of use cases [40]. Instead of modeling malicious behavior by way of use cases, they modify normal use cases to fulfill security requirements that were specified before. The security requirements specification happens textually, and the use case modifications are realized using stereotypes on top of conventional UML use case notations, so no security-specific visual notations are introduced. This work is closely related to UMLsec, which was discussed earlier.

A similar approach is followed by **Vasilevskaya**, who also proposes to model security enhancements into UML diagrams, but mainly uses UML activity diagrams for visualization [41]. Since the scope is embedded systems, the diagrams model the internal behavior of suchlike systems

very specifically. Again, standard UML notations are used with no additions.

Mellado et al. combine the ideas of misuse cases representing attack scenarios (''invade privacy'') and security use cases representing security requirements (''ensure privacy'') [42]. They use standard UML use case notations, but different colors for normal use cases, misuse cases, and security use cases.

Vivas et al. build their security engineering approach on modeling business processes in UML sequence diagrams [43]. Since only the business processes (entity, relation and function in the concept dictionary), but not their security analysis is modeled, again no security-specific visual notation is needed.

The security concept dictionary coverage (**requirements R1.x**) is summarized in Table 10 as part of the chapter evaluation summary. UML-based languages can represent the function concept with use cases, activity diagrams, or sequence diagrams, and modified versions of use cases represent security-enhanced functions. Security stereotypes representing security requirements and properties reflect the security parameters concept. Unwanted events and attack scenarios are represented via abuse or misuse cases – their level of detail determines if it reflects an unwanted event representing the impact of an attack or a attack scenario modeling the detailed course of attack.

All approaches to integrate security concepts into UML have in common that they use the existing UML language, sometimes extended textually, but do not introduce new visual notations. Therefore, the cognitive effectiveness for UML security extensions is the same as for UML in general. The fulfillment of the cognitive effectiveness requirements (**R2.x**) is listed in Table 3. An X means the requirement is fully met, (x) marks a partially met requirement, and an empty cell means the requirement is not met.

B. SYSML SECURITY EXTENSIONS

The System Modeling Language(SysML), like UML, is maintained and published by OMG [44]. It builds upon UML, but was extended to be used in systems engineering, thus representing not only software, but also physical systems. SysML redefines the UML class diagram as a block definition diagram to reflect the need to model physical structures, and adds two diagram types: the parametric diagram and the requirement diagram (Fig. 5) [45].

Like for UML, there are security extensions for SysML: **Apvrille and Roudier** propose a SysML security extension called SysML-Sec [46]. Like UMLsec, the extension makes use of the extensibility mechanisms provided by UML, which means defining security-specific stereotypes like ≪security requirement≫ or ≪attack≫, which can be mapped to the modeled architecture across all diagram types. Again, the security stereotypes are added to diagrams as textual labels, the only visual representation being specific coloring of the blocks the stereotypes are attached to, and just like UMLsec, SysML-Sec is meant to be used with tools which interpret the diagrams.

Mažeika and Butleris have developed MBSEsec, a security extension for a Model-Based Systems Engineering method that uses SysML for its models [47]. It is similar to SysML-Sec, but defines a larger number of stereotypes. Also, MBSEsec specifies a method which includes guidance on which diagram type is to be used for what purpose in the MBSEsec security engineering process. For example, requirements and use cases diagrams are recommended for security requirement identification. While MBSEsec does not define new visual notations, it does define new diagrams types as extensions of existing UML or SysML diagrams that are better suitable for certain security engineering activities. Examples are the misuse diagram as proposed by Sindre and Opdahl (Fig. 4) or a threat and risk definition diagram, extending the UML class diagram, for modeling threats and risks (Fig. 6).

Lemaire et al. propose a SysML extension specifically designed for the security analysis for Industrial Control Systems (ICS) [7]. Their method includes creating a human-readable system model, represented with a SysML internal block diagram, and it does not contain any security extensions. Its purpose is to be fed to a tool that automatically presents security vulnerabilities and suggested security requirements to the user. The SysML extension addresses the formal reasoning carried out by the tool to derive vulnerabilities and security requirements from the block diagram it is fed. Since these extensions are not meant to be human readable, no new visual notations are introduced. Thus, in terms of the concept dictionary, only the entity, relation and function part is visually represented by Lemaire et al.

The IEEE Power and Energy Society is currently working on a **standard P2808** to diagram security-relevant information for power systems. The working group creating the standard (it is still under development) currently considers designing their drawings based on SysML. Using internal block diagrams to represent components (e.g. a relay) in a power plant or substation network, the idea is to introduce acronyms for categories that mark security-relevant information and display the information values in the respective fields. Examples for this security-relevant information are logging, LAN, cyber alarms, electronic access, block and allow listings, certificate management etc,

The overall coverage of the concept dictionary by SysML-based visual languages is again shown in Table 10: When use cases diagrams are used, SysML can represent the function concept like UML. The SysML block diagrams are additionally used to represent entities and relations (i.e. system hardware and network connections). Security parameters are given in requirement stereotypes or in more detail in the P2808 configuration listings. Attack scenarios are represented by the threat, and risk stereotypes in MBSEsec or the attack stereotype in SysML-Sec, and attack indicators are represented by the vulnerability stereotype in MBSEsec.

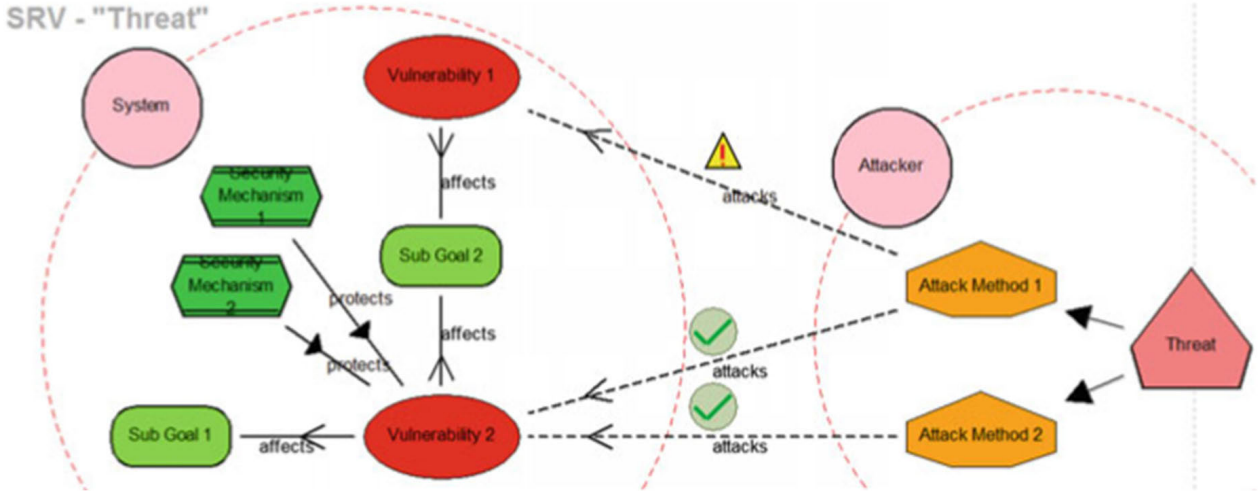


FIGURE 11. Secure Tropos for cloud environments, with additional coloring in visual notations as well as new concepts "vulnerability" and "attack method" [53].

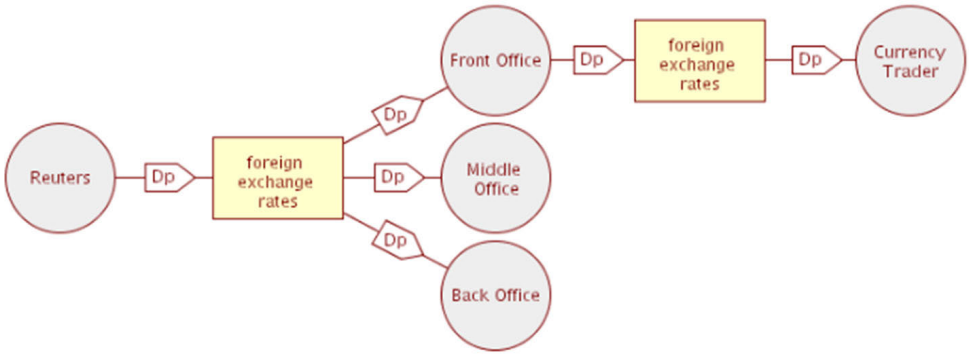


FIGURE 12. Secure Tropos diagram containing "delegation of permission" dependencies between actors (circles) and resources (rectangles) [54].

Because UML and SysML are so similar, use the same diagram types and visual vocabularies, their cognitive effectiveness is the same. Therefore, the evaluation of the cognitive effectiveness requirements for UML (Table 3) also applies to SysML.

C. TROPOS AND I* SECURITY EXTENSIONS

The i* framework (pronounced eye star) was developed by Eric S. Yu as an attempt to include social aspects – humans and their intentions – into information system engineering methods [48]. The central concepts of the i* framework are actors, goals, tasks, resources, and dependencies. Actors have intentions, represented by their goals. They may be dependent on others helping them achieve their goals, carrying out a task or providing a resource. There are also softgoals in i*, which represent qualities – for example ''secure''.

Because actors and their intentions are the core concept of i*, it is called ''agent-oriented''. There is an agent-oriented software engineering method called Tropos [49], which was inspired by i* and uses it as its modeling framework. The visual notations for the central i* concepts are shown in Fig. 7.

Although not developed specifically for that purpose, the inclusion of humans in the modeling is an important precondition to accurately model security aspects of systems as well, since attackers are humans. On top of that, human interactions make up a big portion of systems' vulnerabilities, but also security countermeasures. Consequently, security extensions of the Tropos methodology, called Secure Tropos, have been proposed.

Mouratidis' and Giorgini's Secure Tropos [50] adds security constraints, ''secure'' versions of actor, task and resource, and the new concepts security feature, protection objective, security mechanism, and threat (Fig. 8, Fig. 10). If compared to the original Tropos / i* visual language, only the threat is a newly introduced shape. The others use existing shapes, which gain an additional meaning.

Asnar et al. structure the Tropos methodology in layers in a ''goal risk model'' [51]: In the first layer, goals are defined, in the second, events (risks) are introduced and in the third, treatments to prevent or mitigate these events are defined. The visual notations are consistent with the Secure Tropos notations: ellipses for goals, pentagons (like the ''threat shape'' in Secure Tropos) for events, and hexagons for treatments.

Mayer et al. also add visual notations for threats and vulnerabilities, but they use diamond shapes for threats and the same with a black corner for vulnerabilities [52].

Later, **Mouratidis** et al. extended their Secure Tropos language for application to cloud environments [53]. Not only do

TABLE 5. Cognitive effectiveness of CORAS.

ID	Requirement	Met?	Comment
R2.1	**One symbol** per concept, one concept per symbol	X	Fulfilled.
R2.2	**Limited (<10) number** of different symbols	X	Eight symbols.
R2.3	Use of **unique icons** plus additional visual variables for discrimination	X	Unique icons plus different coloring.
R2.4	Use of **text** for additional information or as an interpretation reminder, but not for discrimination	X	Text is used for describing elements in more detail.
R2.5	**Collision avoidance** with common visual concepts from automation engineering	X	No icon is commonly used in automation engineering.
R2.6	**Modularized diagrams** with limited number of elements	(x)	There are examples for modularized diagrams for certain purposes (e.g. calculating risk likelihood), but no guidance is given how to modularize.
R2.7	**Cognitive integration** mechanism between diagrams		No relation between diagrams visible.

TABLE 6. Cognitive effectiveness of SecVDSL.

ID	Requirement	Met?	Comment
R2.1	**One symbol** per concept, one concept per symbol	X	
R2.2	**Limited (<10) number** of different symbols	X	Ten symbols (number of symbols is at the upper limit for cognitive effectivebess)
R2.3	Use of **unique icons** plus additional visual variables for discrimination	(x)	Unique icons. Coloring is no additional means for visual discrimination, but conveys additional information (e.g. criticality for assets).
R2.4	Use of **text** for additional information or as an interpretation reminder, but not for discrimination	X	Text is used for concepts' titles.
R2.5	**Collision avoidance** with common visual concepts from automation engineering	X	No icon is commonly used in automation engineering.
R2.6	**Modularized diagrams** with limited number of elements	(x)	Modularization means that only one concept per diagram is shown. This makes all efforts for visual discrimination useless because icons are never present in the same diagram. Also, it limits the usefulness of the diagrams.
R2.7	**Cognitive integration** mechanism between diagrams		No relation between diagrams visible, authors mention drill-down mechanism between diagrams in the tool as an integration mechanism.

they add color as an additional visual variable to distinguish between visual notations, but they also add multiple new concepts like vulnerability and attack method (Fig. 11).

Massacci's and Zannone's additions to Secure Tropos [54] consist in mapping security-relevant concepts to dependencies: ownership, trust and delegation. The new visual notations to represent these concepts are labels with letters - T for trust, S for distrust, O for ownership, D for delegation-, extended by an additional letter e for execution and p for permission (Fig. 12). For example, a "Dp" dependency stands for delegation of permission, which means the transfer of certain rights from one entity to another.

Another approach to add security concepts to Tropos is proposed by **Elahi and Yu** [55]: They add malicious versions to existing concepts, visually distinguishable by black backgrounds (Fig. 9). The counterpart for an actor is a malicious actor, and a threat or attack is composed of malicious versions of goals and tasks. Additionally, a black dot is specified to represent vulnerabilities.

The concept dictionary coverage for i*/Tropos-based visual languages is summarized in Table 10 as part of the evaluation summary. Parts of the function concept may be represented using dependencies. Security parameters are represented by treatments, constraints, or security mechanisms, as well as the "security" versions of tasks and resources. For access control security parameters, the ownership, trust and delegation dependencies can be used. Security goals are covered via security goals or soft goals, attack scenarios via events, threats, attacks, attackers, malicious actors and attack methods, and attack indicators via vulnerability (points).

In Table 4, the cognitive effectiveness requirements are evaluated for the i*/Tropos-based visual languages introduced in this section.

D. CORAS

Next to the approaches that extend existing visual languages to also model security, there are some approaches that develop completely new visual languages for security. One of those is **CORAS**, published by **Lund et al.** [56]. CORAS introduces a set of symbols (Fig. 13) and diagrams for security risk analyses. All semantics are covered by the icons, the relations between icons are represented by simple arrows without

TABLE 7. Cognitive effectiveness of the discussed network diagrams.

ID	Requirement	Met?	Comment
R2.1	**One symbol** per concept, one concept per symbol	X	Fulfilled in most network diagrams.
R2.2	**Limited (<10) number** of different symbols		It's easy and common to exceed 10 different elements in a network diagram alone. And this does not yet include any additional security concepts that would need a symbol as well.
R2.3	Use of **unique icons** plus additional visual variables for discrimination	(x)	Icons are common, but additional visual variables are rarely used.
R2.4	Use of **text** for additional information or as an interpretation reminder, but not for discrimination	X	Text is used for labeling only.
R2.5	**Collision avoidance** with common visual concepts from automation engineering	X	At least if icons are used, they do not collide with automation engineering diagrams.
R2.6	**Modularized diagrams** with limited number of elements	(x)	Network diagrams tend to be split by location, sub network, etc.
R2.7	**Cognitive integration** mechanism between diagrams	(x)	If diagrams are split, sometimes the "neighboring" diagram is indicated with arrows to the side, or an overview diagram shows which areas other diagrams "drill down" into.

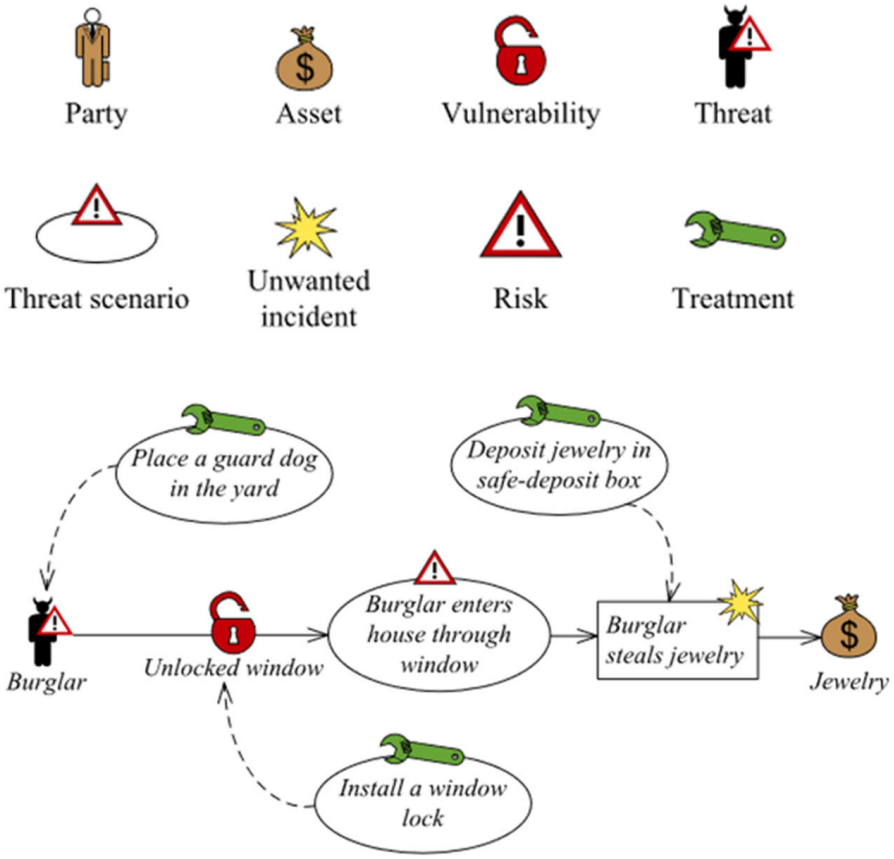


FIGURE 13. Symbols defined in CORAS (top) and example CORAS diagram including asset, unwanted incident, risk, vulnerability, threat, and treatments (bottom) [56].

specific semantics (see example diagram in Fig. 13). On top, UML activity diagrams are used.

Like for the other evaluated evaluations, the concept dictionary coverage of CORAS is given in Table 10. CORAS only displays the system under consideration as assets, which could give indications for entities and relations. Security parameters are represented as treatments, unwanted events as unwanted incidents, attack indicators as vulnerabilities, and attack scenarios as a combination of threat, threat scenario, and risk.

Table 5 summarizes the evaluation of the cognitive effectiveness requirements for the CORAS visual language.

E. SECDSVL

Almorsy and Grundy propose a Domain-Specific Visual Language for Security, called **SecDSVL** [57]. Like CORAS,

TABLE 8. Cognitive effectiveness of the discussed data flow diagrams.

ID	Requirement	Met?	Comment
R2.1	**One symbol** per concept, one concept per symbol	X	
R2.2	**Limited (<10) number** of different symbols	X	Four symbols in the versions by Kozar [61] or Drewry [62]. Often, realized simply as textual labels on relations in network diagrams.
R2.3	Use of **unique icons** plus additional visual variables for discrimination		No icons used.
R2.4	Use of **text** for additional information or as an interpretation reminder, but not for discrimination	X	
R2.5	**Collision avoidance** with common visual concepts from automation engineering	(x)	Ellipses specified by Drewry collide with PCE requests. Process boxes from the same specification resemble blocks in sequential function charts (SFCs).
R2.6	**Modularized diagrams** with limited number of elements		Not mentioned.
R2.7	**Cognitive integration** mechanism between diagrams		Not mentioned.

the language includes icons for all ten concepts it represents (Fig. 14): asset, security objective, threat, threat agent, vulnerability, attack, security requirement, security zone, security service, and security control. No diagram types are explicitly defined, but examples for diagrams based on the defined icons are given. Based on the examples, most diagrams include only one icon category (Fig. 15).

The concept dictionary coverage of SecDSVL is again summarized in Table 10. Like CORAS, SecDSVL at most give hints towards entities and relations in the form of interconnected asset symbols. Security parameters are represented by the icons for security requirements, zones, services and controls. Security goals are called security objectives in SecDSVL. Attack indicators are represented by security vulnerabilities, and attack scenarios by a combination of security attacks, threat agents, and security threats.

Almorsy and Grundy have taken into account Moody's principles of cognitive effectiveness, that were the basis for the cognitive effectiveness requirements as well, for creating their visual language. Therefore, it comes at no surprise that many of the cognitive effectivess requirements are met at notation level. However, SecVDSL has weaknesses at diagram level. An overview of the language's fulfillment of cognitive effectiveness requirements is given in Table 6.

F. NETWORK DIAGRAMS

So far, those visual languages for security were discussed that address the entire security decision-making process: understanding the system to be protected, risks, goals and requirements, and solutions.

However, there are also visual languages that cover only parts of security decision-making. Diagrams that aid understanding of the system under consideration are network diagrams and data flow diagrams. They are covered in this and the following section.

The architecture (entities and relations in the concept dictionary) of the system under consideration is often represented in network diagrams. These are non-standardized drawings using icons to represent the technical components in a network (router, switch, workstation, server, ...). Fig. 16 shows an example screenshot [58] from the **Cybersecurity Evaluation Tool (CSET)** [13] provided by the US cybersecurity and infrastructure security agency (CISA) to aid critical infrastructures identifying security requirements. The sample diagram is simple, but depending on network size, these diagrams can grow considerably large and complex.

If network diagrams are used for security purposes, their elements are sometimes grouped in security zones, marking assets with similar properties leading to a similar criticality. The concepts of zones and conduits (which are the communication channels between zones) stems from IEC 62443-3-2 [59].

The concept dictionary coverage of network diagrams is narrow, as remarked in the introduction: Solely the entities and relations are represented (Table 10). The cognitive effectiveness requirements for the discussed network diagrams are evaluated in Table 7.

G. DATA FLOW DIAGRAMS

Data flow diagrams (DFDs) are common, and like network diagrams, not always in a standardized notation. In the concept dictionary, they represent function.

Sometimes, DFDs simply consist in adding data flows to network diagrams as discussed in the last section. Some DFDs resemble or build upon UML notations, like the deployment diagram including communication interactions shown in **Lipner's and Howard's** description of the **Microsoft Security Development Lifecycle (SDL)** [60].

There are also attempts to define specific notations for data flow diagrams, presented by **Kozar** [61] or **Drewry** [62].The notations as well as an example data flow diagram are given in Fig. 17.

Like for network diagrams, the concept dictionary coverage of data flow diagrams is narrow: Solely the function concept is represented (Table 10). The presented data flow diagrams' cognitive effectiveness requirements are evaluated in Table 8.

FIGURE 14. Icons defined in SecDSVL [57].

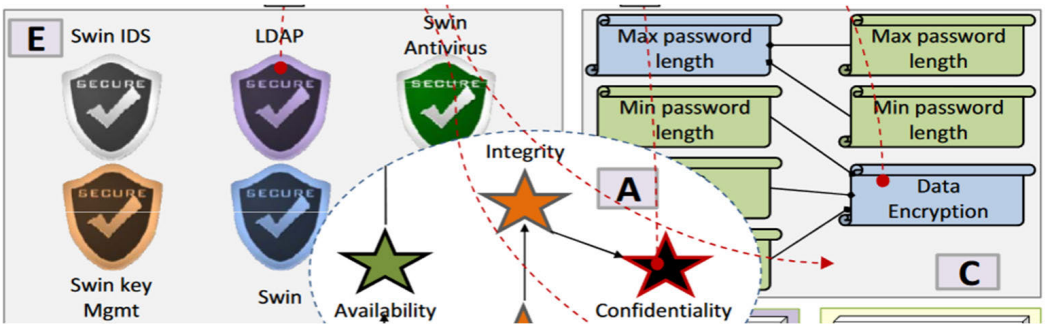


FIGURE 15. Diagram examples from SecVDSL [57]: Security control diagram (left), security objective diagram (center), security requirements diagram (right).

TABLE 9. Cognitive effectiveness of the discussed attack tree diagrams.

ID	Requirement	Met?	Comment
R2.1	**One symbol** per concept, one concept per symbol	X	
R2.2	**Limited (<10) number** of different symbols	X	The number of concepts is very limited, since attack trees / graphs only represent possible attacks and are no visual language for the full security engineering process.
R2.3	Use of **unique icons** plus additional visual variables for discrimination		Across all attack tree / graph variants, only shapes (not icons) and color are used for discrimination.
R2.4	Use of **text** for additional information or as an interpretation reminder, but not for discrimination	X	
R2.5	**Collision avoidance** with common visual concepts from automation engineering	(x)	Ellipses resemble PCE requests. Graphs as a whole resemble sequential function charts, but are not really at risk for being confused since the areas of application would be very different.
R2.6	**Modularized diagrams** with limited number of elements	(x)	Attack trees can be modularized by splitting them up by attack goal. Nevertheless, they tend to become very complex because their idea is to represent all possible attack scenarios.
R2.7	**Cognitive integration** mechanism between diagrams		If modularized, diagrams are not regarded as connected.

TABLE 10. Evaluation summary for the semantic requirements (security concept dictionary coverage, R1.x).

Concept dictionary coverage requirements (R1.x)	UML-based	SysML-based	i*/Tropos-based	CORAS	SecDSVL	Network diagrams	Data flow diagrams	Attack tree diagr.
R1.1 Entity and relation		X		(x)	(x)	X		
R1.2 Function	X	X	(x)				X	
R1.3 Security parameter	X	X	X	X	X			(x)
R1.4 Security goal			X		X			(x)
R1.5 Unwanted event indicator								
R1.6 Unwanted event	X			X				X
R1.7 Attack indicator		X	X	X	X			(x)
R1.8 Attack scenario	X	X	X	X	X			X

H. ATTACK TREE DIAGRAMS

Like network diagrams and data flow diagrams for the system under consideration, there are also diagrams that only help with modeling and understanding security problems, more concisely, security attacks. They are usually graph diagrams, called attack graphs or attack trees. Not all attack tree dia-

TABLE 11. Evaluation summary for the visual syntax requirements (cognitive effectiveness, R2.x).

Cognitive effectiveness requirements (R2.x)	UML-based	SysML-based	i*/Tropos-based	CORAS	SecDSVL	Network diagrams	Data flow diagrams	Attack tree diagr.
R2.1 One symbol per concept, one concept per symbol			(x)	X	X	X	X	X
R2.2 Limited (<10) number of different symbols			(x)	X	X		X	X
R2.3 Use of unique icons plus additional visual variables for discrimination				X	(x)	(x)		
R2.4 Use of text for additional information or as an interpretation reminder, but not for discrimination			(x)	X	X	X	X	X
R2.5 Collision avoidance with common visual concepts from automation engineering	(x)	(x)		X	X	X	(x)	(x)
R2.6 Modularized diagrams with limited number of elements	(x)	(x)		(x)	(x)	(x)		(x)
R2.7 Cognitive integration mechanism between diagrams						(x)		

grams are intended to be read by humans. Some are part of a formal notation that is meant to be interpreted and analyzed by an algorithm. In this section, only those attack graphs are introduced that are to be interpreted by humans.

Attack trees were first introduced by Schneier in 1999 [63], and first adopted for ICS by Byres et al. five years later [64]. The principle is simple: For every attack goal, there is a separate attack tree. The tree consists of all possible attacks to achieve the attack goal and thus helps so systematically think through all attack possibilities. Within the concept dictionary, attack graph diagrams represent unwanted events (as attack goals) and attack scenarios (a path through the attack tree); sometimes also attack indicators. An example from Byres et al. is shown in Fig. 18. All attack goals have the same shape, however, their severity is epressented by different coloring. Hoff [65] uses a similar principle by representing different MITRE ATT&CK for ICS [66] tactics (from initial attacks to final impact on target) using a color gradient.

The ''obstacle trees'' in the anti-goal model introduced by **van Lamsweerde** [67] look similar. Rhomboid-shaped ''Anti-goals'' are negative impacts on the system under consideration. Obstacle trees add an additional shape for ''vulnerability'' though: a pentagon.

LeMay et al. introduce even more different concepts in their ''attack execution graphs'' [8]: For each attack step (rectangle) the goals are displayed in ellipses, and needed access, skills, and knowledge in squares, circles, and triangles. All concepts are additionally differentiated by color (Fig. 19).

Kordy et al. [68] make two additions to the basic attack tree concept in their ''attack defense trees''. Both are also visually represented (Fig. 20): First, they distinguish between disjunctive and conjunctive refinements of objectives in their attack trees. Disjunctive refinements are alternatives to reach the parent objective, conjunctive refinements are both needed to achieve the parent objective, marked by a connecting line between their edges. Second, they introduce defense nodes, which mark possible defenses against the attack nodes they're connected to. That way, they also include the security requirements (e.g. security parameters set to certain values) from the concept dictionary.

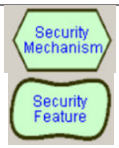
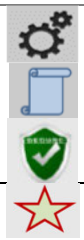
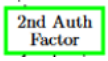
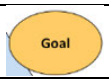
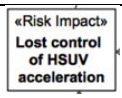
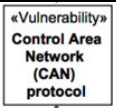
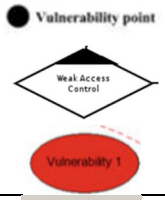
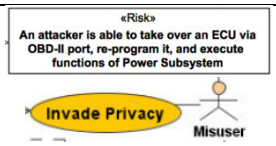
Again, the concept dictionary coverage of attack tree or graph diagrams in their basic version is relatively narrow, as summarized in Table 10: Depending on their level abstraction, they cover unwanted events (i.e. impacts on the system) or attack scenarios (single attack steps leading to these impacts). Concept coverage is extended depending on the attack tree diagram variant: Attack defense trees by Kordy et al. [68] additionally represent security parameters through their countermeasures. Attack execution graphs by LeMay et al. [8] additionally represent security goals. Obstacle trees by van Lamsweerde [67] additionally represent attack indicators through their vulnerabilities.

The evaluation of the cognitive effectiveness requirements for the discussed attack trees or graphs is given in Table 9. Like network and data flow diagrams, attack graphs only represent a small portion of the concepts needed for a security engineering visual language, so differentiating between these concepts is not too hard.

VI. EVALUATION SUMMARY

Table 10 and Table 11 summarize the evaluation across all requirements and all reviewed visual notations. In both tables, X means the requirement is fully met, (x) marks a partially

TABLE 12. Visual notations for the concepts in the concept dictionary (images from sources evaluated in chapter).

Concept dictionary coverage requirements (R1.x)	UML-based	SysML-based	i*/Tropos-based	CORAS	SecDSVL	Attack tree diagr.
R1.3 Security parameter						
R1.4 Security goal						
R1.6 Unwanted event						
R1.7 Attack indicator						
R1.8 Attack scenario						

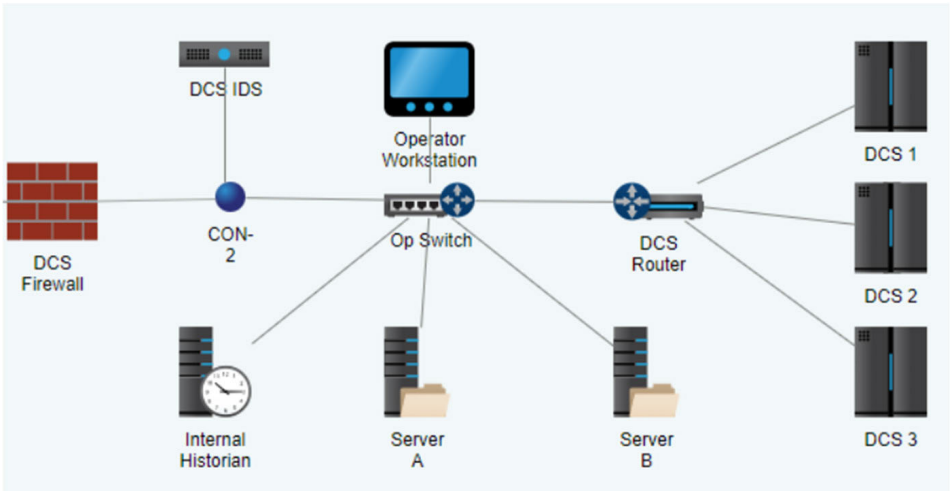


FIGURE 16. Exemplary network drawing in the CSET tool [13] (screenshot from [58]).

met requirement, and an empty cell means the requirement is not met. The key findings are briefly touched upon in the following, beginning with the evaluation of the semantic requirements – more precisely, the concept dictionary coverage.

A key finding of the authors' evaluation is presented in Table 10: no visual notation covers all concepts. The lack of notations mostly pertains to the concepts at the intersection of system and security domain knowledge – unwanted event indicators and security parameters.

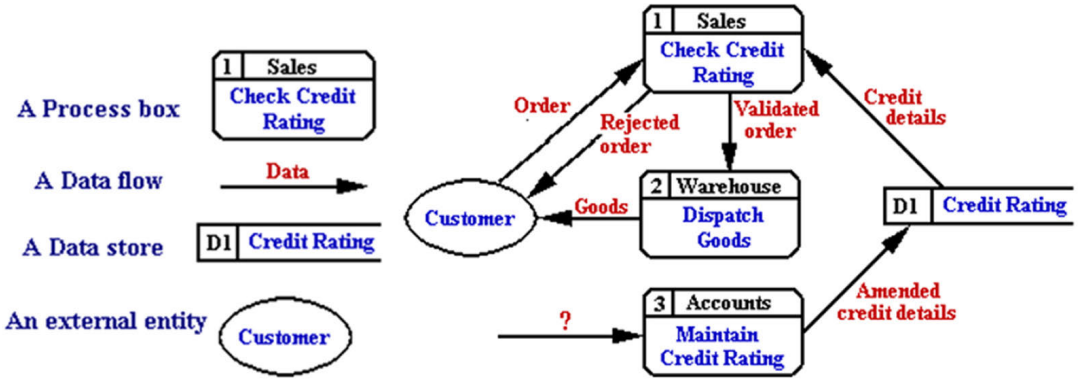


FIGURE 17. Data flow diagram notation as specified by Drewry [62].

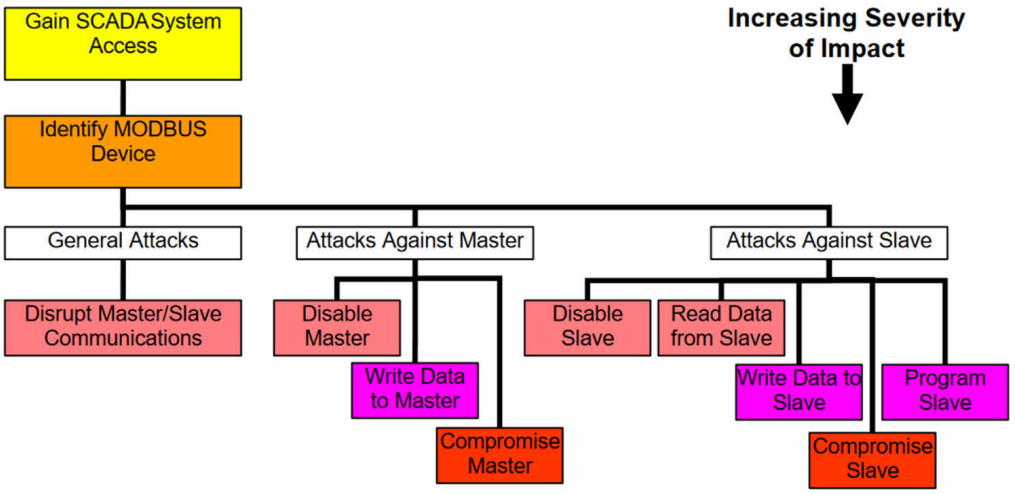


FIGURE 18. Attack tree for ICS by Byres et al. [64]. The colors represent the severity of impact.

Unwanted event indicators are not visualized in any of the reviewed diagrams. This does not come as a surprise, because the unwanted event indicator is a concept that marks security-relevant aspects in information about the system under consideration developed by other domains. This is rarely covered by existing languages.

The same applies to security parameters. The only reason some of the reviewed notations are evaluated to visualize security parameters is that a broad definition of security parameters has been applied, also covering security requirements and measures.

Concepts that are purely in the security domain (attack indicator, attack scenario) are covered in more approaches than concepts that are rather in the domain of the system under consideration (unwanted event, function). This is not surprising since most visual languages in the context of security aim at collecting primarily security domain knowledge. However, it is striking that the security goal, a relatively common concept, is only represented in two of the visual languages.

This can be summarized as **key finding a): Concepts at the intersection of system and security knowledge (security parameters, unwanted events, unwanted event indicators) are not well represented in existing languages.** For these concepts, improved notations need to be developed. For all other concepts, there are existing notations which may be adopted.

Table 12 shows an overview of notations for those concepts which were covered by the reviewed languages (the unwanted event indicator wasn't at all). Also, the entity, relation and function concepts are excluded since they are similar across all languages, represented by use-case-like diagrams, and these cannot be condensed into one symbol. For the same reason, the network and data flow diagrams are not included in Table 12.

Table 12 serves as an illustration for the findings from the cognitive effectiveness evaluation summarized in Table 11. Already at first sight, Table 11 conveys the impression that the specialized notations (CORAS, SecDSVL, attack tree diagrams) have a better cognitive effectiveness than those that adopt more general visual notations like UML, SysML or i*/Tropos for security, and this is underlined by the fact that the specialized notations address almost all visual syntax requirements from Table 11, while the general language extensions barely address any of them.

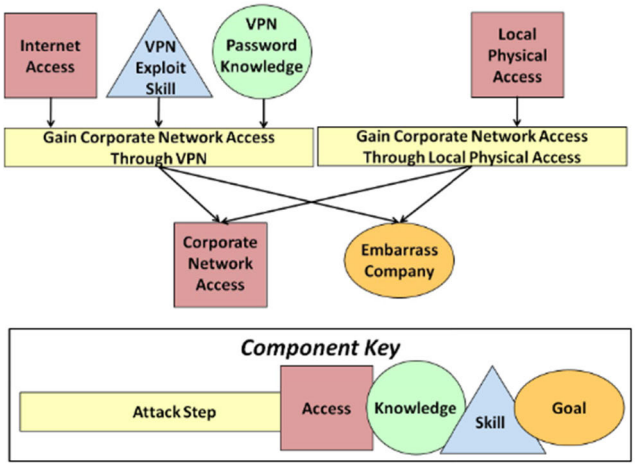


FIGURE 19. Attack execution graph by LeMay et al. [8] including representation of access, knowledge and skill next to attack steps and their goals.

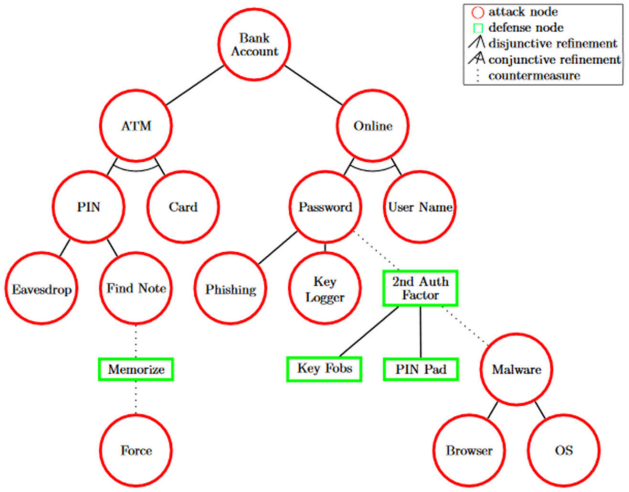


FIGURE 20. Attack Defense Tree by Kordy et al., including defense nodes (green rectangles) representing countermeasures against attack nodes (red circles).

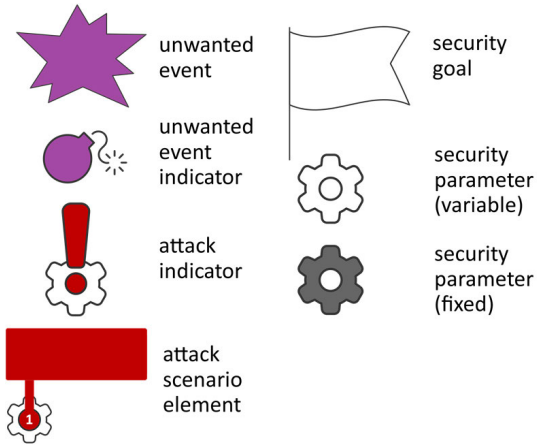


FIGURE 21. Icons of the proposed visual language for security design decision-making. Coloring is used to group icons by diagram type: unwanted event diagram (violet), security decision diagram (white / grey), attack diagram (red).

This is because despite their popularity, these general languages already have a rather weak cognitive effectiveness in their fundamental notations, as analyzed in [69] for UML: They rely exclusively on shape to distinguish between concepts, and the shapes they use are plain geometric shapes which have to be learned like vocabularies to be understood. The extensions for UML and SysML are even worse, since they rely only on text added in guillemets (see first two columns in Table 12). I*/Tropos (third column) is only slightly better. It uses different geometric shapes and sometimes adds color, and extensions introduce new shapes – but again geometric ones.

This makes for **key finding b): general-purpose visual languages like UML, SysML, or i*/Tropos are not an effective basis for a visual language for ICS security decision-making because their cognitive effectiveness is weak.** Extending UML, SysML or i*/Tropos can be a good compromise despite their weak cognitive effectiveness if the target group of the visual language is very familiar with these languages. However, the target group being ICS engineers and not software engineers, this advantage is non-existent in this work. Therefore, based on the evaluation, developing a specialized visual language for ICS security engineering seems more reasonable.

Taking a closer look at specialized notations, they are either complete languages specifically created for security (CORAS and SecDSVL) or specialized diagrams which only visualize one of the required concepts (network, data flow, and attack tree diagrams).

The specialized notations overall have a good coverage of requirements affecting the individual notations (**R2.1 – R2.5**). This is mostly due to the fact that they have a limited set of symbols – the main advantage of the specialization – and make use of icons which are more intuitively understood because they have ''semantic transparency'', to use Moody's wording [20]. Some of the icons have a thematic overlap, which gives hints to symbols that could be chosen in an improved visual language: ''something to adjust'' (wrench, cogwheels) for security parameters, ''something explosion-related'' (explosion, bomb) for the unwanted event, ''something to be wary of'' (warning sign, red ''x'', evil-looking figure) for attack scenario. This can be summarized as **key finding c): Use of icons is a main driver for cognitive effectiveness because of their intuitiveness (''semantic transparency''). There are no established icons, but for some concepts, thematic preferences can be observed.**

However, the specialized notations do not excel at the requirements at diagram level: **R2.6** (modularized diagrams with limited number of elements) and **R2.7** (cognitive integration mechanism between diagrams).

This means the visual notations sacrifice their otherwise good cognitive effectiveness of their individual notations once they are turned into diagrams: They either overwhelm viewers with diagrams growing to unlimited size and complexity by not having any built-in modularization options,

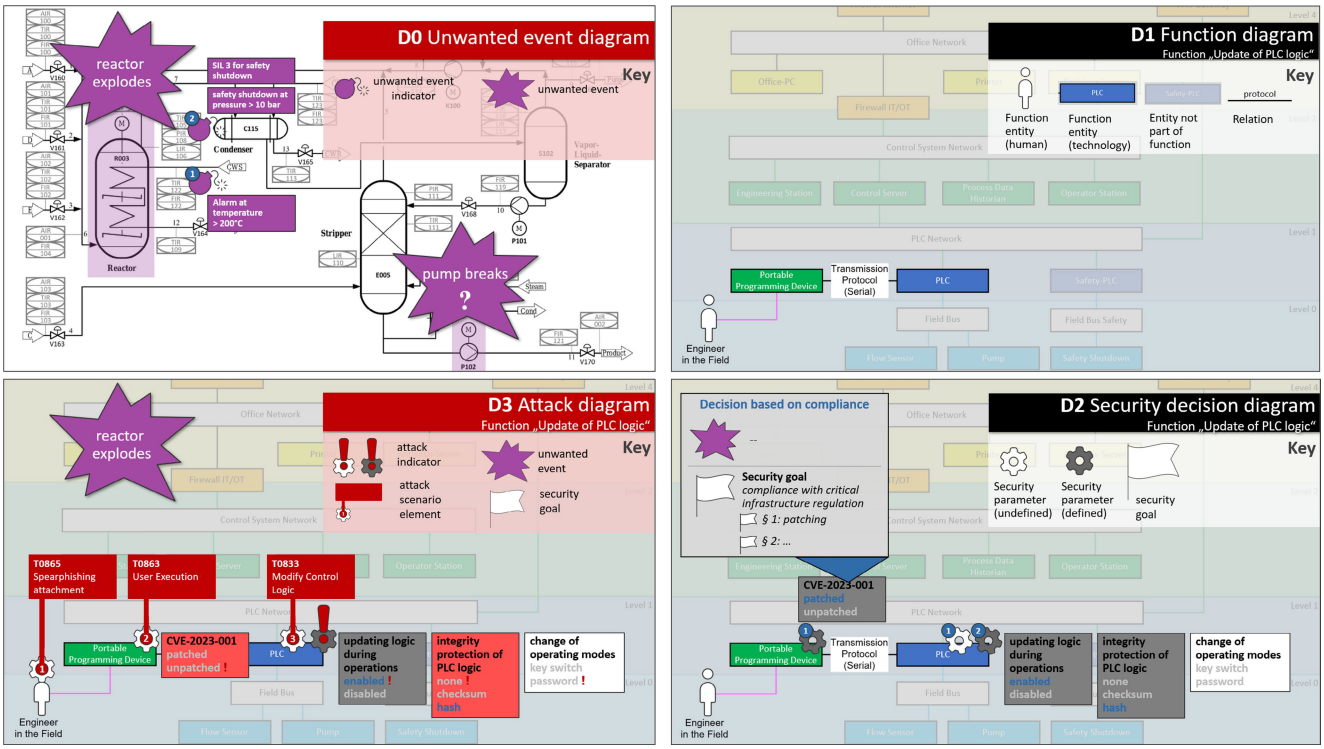


FIGURE 22. Diagram types of the envisioned visual language for security design decision-making: Unwanted event diagram, function diagram, security decision diagram, attack diagram.

or they leave viewers with information scattered over multiple diagrams with no built-in guidance to integrate all this to a greater whole.

This leads to **key finding d): a good visual language needs to include rules on diagram level: built-in guidance on both effective modularization of diagrams and integration mechanisms for the resulting "diagram network".**

VII. TOWARD A VISUAL LANGUAGE FOR SECURITY DESIGN DECISION-MAKING

The conclusion of the evaluation can be summarized as follows:

- There is a need for a better visual language for security decision-making during the design of ICS. This visual language should not be the extension of a general-purpose language, but a specialized language for security **(key finding b).**
- To achieve good cognitive effectiveness on diagram level, the language needs to provide built-in "filters" to reduce diagram complexity: rules to modularize diagrams containing and mechanisms for the easy integration of the resulting "diagram network" **(key finding d)**.
- To achieve a good cognitive effectiveness on visual vocabulary level, the language should include icons for each represented concept. For security-related concepts, thematic preferences from existing visual notations should be adhered to **(key finding c)**.
- For concepts at the intersection of system knowledge and security knowledge (security parameters, unwanted events, unwanted event indicators), new icons need to be created **(key finding a)**.

Equipped with these findings, the main pillars for an improved visual language for security engineering can be defined. Additionally, the fulfillment of the semantic requirements (**R1.x**) and cognitive effectiveness requirements (**R2.x**) defined in chapter IV is indicated.

Pillar one: Consistent basic diagrams as an integration mechanism. All diagrams represent the system under consideration using established entity-relation diagrams like network maps and P&IDs. This covers the concepts entity and relation (**R1.1**) and serves as an integration mechanism between modularized diagrams (**R2.7**), because it means all diagrams have the same basic structure (see Fig. 22).

Pillar two: Functions as a modularization mechanism. Functions (**R1.2**) are visualized by highlighting relevant parts of the basic entity-relation diagrams and adding security-relevant elements from data flow diagrams and use cases: communication protocols and human interactions (see function diagram in Fig. 22) [70]. They are used as one mechanism to modularize diagrams (**R2.6**): For each function, a different part of the entity-relation diagram is highlighted. This not only makes diagrams less complex (**R2.2**) but also aids security decision-making because attack scenarios and security parameters can be looked at function by function, asking how manipulating or disabling the function can lead to an unwanted event.

Pillar three: Icons for semantic transparency. All other concepts (**R1.3 - R1.8**) are represented as icons (**R2.1, R2.3**), taking into account the thematic preferences found in the visual language evaluation: The security parameters (thematic preference: ''something to adjust'') are represented by cogwheels. Security goals, for which no thematic preference was observed in existing visual languages, are represented by a (finishing) flag. Attack indicators, which mark critical values of the security indicators are represented by the cogwheel enhanced by ''something to be wary of'' – an exclamation mark. Attack scenarios build upon these attack indicators. Unwanted events and unwanted event indicators make use of the ''something explosion-related'' theme: Unwanted event indicators, which are pieces of information that could, but don't have to, point to a negative consequence, are represented by bombs (which could or could not explode). Unwanted events however mark consequences that are definitely confirmed to be negative, so they are represented by explosions. An overview of icons is given in Fig. 21. All icons can be layered on top of the basic diagrams (see Fig. 22).

Pillar four: Diagram types as a modularization mechanism. Not every icon is needed in every diagram. In addition to the modularization by function, diagrams are modularized (**R2.6**) by how they can be used in the security design decision-making workflow. Each diagram is designed to assist during one (or more) steps in workflow, and only the relevant icons for this step are displayed. Thus, four diagram types are proposed (Fig. 22): Unwanted event diagrams, function diagrams, security decision diagrams, and attack diagrams. As a second visual differentiator, color is used to group related icons. Icons that belong onto the same diagram type (Fig. 21) have the same color: unwanted event diagram (violet), security decision diagram (white / grey), or attack diagram (red).

The visual language is designed to support the security decision-making workflow during the design of an ICS. Table 13 provides an overview which steps in the security design decision-making workflow are supported by which of the diagram types. As analyzed in chapter III, for each workflow step, visualizations support understanding the relevant input information, making the necessary decisions as well as documenting them.

It is important to note that the visual language will not be practical if the diagrams are drawn manually, using pen and paper or a drawing software.

The modularization mechanisms (separate diagrams for each function and diagram type), which are paramount for ensuring the diagram's cognitive effectiveness (**R2.6, R2.7**) cause a lot of slightly different diagram variants. This is not a problem if a software tool generates the multiple diagrams from a single electronic data model - but for manually drawn models, version management would be a nightmare.

Also, the icons layered on top of basic diagrams rely on dynamic filtering mechanisms and on the ability to provide additional information on demand. This is good practice for electronically generated diagrams in software tools, but again would not be feasible on manually created diagrams.

TABLE 13. How diagrams support the security design decision-making workflow.

ID	Security Decision-Making Workflow	Supporting Diagram
1	Identify **unwanted events**	**D0 Unwanted event diagram**
2	Identify the essential **entities, relations** and **functions** of the system under consideration	**D1 Function diagram**
3	Identify variable **security parameters**	**D2 Security decision diagram**
4	Identify **attack indicators** and **attack scenarios** to determine how system functions and security parameters can contribute to unwanted events	**D3 Attack diagram**
5	Identify **security goals** to prevent unwanted events	any of the diagrams
6	Set security parameters' values to meet security goals	**D2 Security decision diagram**

Therefore, there is a fifth pillar for the proposed visual language concept:

Pillar five: Electronic data model and software tool. The visual language must include an electronic data model and a software tool to guide through the diagram-supported security decision-making workflow, and to generate and manipulate the visualizations dynamically.

This also opens the possibility to include the diagrams created during security design decision-making as part of the control system HMI and / or SOC visualizations used for system operations, making it easier to reference and update security design diagrams throughout the systems' life span.

VIII. CONCLUSION AND OUTLOOK

This paper motivates the need and investigates the requirements for a specialized visual language for security decision-making during the design of ICS.

Just like security operations information must be presented to security operators sitting in front of a security operations center (SOC) monitor in a way that enables them to decide how to react to a security incident, security engineering information must be presented to security engineers in a way that enables them to decide how to design a defensible system.

Therefore, as a prerequisite for constructing a visual language for security engineering in the future, this paper explores two scientific questions: *1) what are the requirements for visualizing security-relevant engineering information in a way that enables engineers to make security decisions during ICS design?* and *2) which existing visual languages meet (parts of) these requirements?*

Visualizing information that is relevant for making security design decisions – more precisely, selecting security measures – involves solving a practical problem: Security design decision-making requires understanding and processing large amounts of information, most of which will be outside the area of expertise of any security decision maker, regardless

of their profession. Visual representations have the potential to effectively convey suchlike information, thus saving the security engineers' brain capacity for the decision-making. However, visualizations only unleash this potential if they are carefully constructed for cognitive effectiveness. Therefore, cognitive effectiveness requirements for the visual syntax of a future visual language for security engineering were defined.

The case for using visualizations was further substantiated by outlining how each step in a security design decision-making workflow could benefit from visualizations, also revealing that fundamental portions of the relevant information are commonly visualized anyway. As a result, requirements for the semantics of a future visual language were defined in the form of a concept dictionary.

Next, existing visual notations intended to be read by humans in the context of security decision-making were portrayed and evaluated against the defined requirements. The evaluation led to **four key findings**:

(a) Concepts at the intersection of system and security knowledge (security parameters, unwanted events, unwanted event indicators) are not well represented in existing languages. **(b)** General-purpose visual languages like UML, SysML, or i*/Tropos are not an effective basis for a visual language for ICS security decision-making because their cognitive effectiveness is weak. **(c)** Use of icons is a main driver for cognitive effectiveness because of their intuitiveness (''semantic transparency''). There are no established icons, but for some concepts, thematic preferences can be observed. **(d)** A good visual language needs to include rules on diagram level: built-in guidance on both effective modularization of diagrams and integration mechanisms for the resulting ''diagram network''.

Equipped with these findings, **five pillars for an improved visual language** for security design decision-making were defined: **(1)** The visual language must include consistent basic diagrams, which are commonly used technical drawings (perhaps in more abstract form) representing the system under consideration. The security-relevant engineering information can then be layered onto these basic diagrams either by **(2)** highlighting certain entities (for functions) or by **(3)** adding icons (for all other concepts: unwanted events, unwanted event indicators, security parameters, security goals, attack indicators, and attack scenarios). Both the functions and the other concepts can be used as modularization mechanisms, i.e. as filters that reduce diagram complexity. Thus, **(4)** four different diagram types can be defined that are each tailored to support specific steps in the security design decision-making workflow. To facilitate these filtering mechanisms without the hassle of creating, versioning and updating the multitude of different diagrams, **(5)** the visual language must be supported by an electronic data model and software tool which can dynamically generate the diagrams, display additional detail on demand, and document the decisions that are made including the reasoning behind each decision.

The vision for the future visual language is that it enables security engineers (regardless of their technical background) to consciously make security decisions during the design of ICS. It should help distinguish the security-relevant from the irrelevant in the haystack of engineering information. Its diagrams should show at a glance not only which information matters for security decision-making, but also why.

Additionally, the modularization mechanisms in the envisioned visual language should provide guidance for slicing the overwhelming problem of finding just the right amount of security for large and complex ICS with their functional requirements, financial constraints and regulatory obligations into digestible chunks. First, by looking at the system under consideration function by function. Second, by analyzing each function diagram type by diagram type. As each diagram type should be designed to support a specific step in the security decision-making workflow, they are also intended to guide the decision-maker through the decision-making even if she doesn't memorize all the required
steps.

One question that a visual language alone cannot answer is from where it obtains the contents of the security-relevant information it displays: Who defines all the functions, security parameters, attack indicators, and unwanted event indicators? Clearly, the security decision-maker will have neither the time nor the expertise for that. The answer is outside the scope of this paper, but because of its fundamental importance for the effectiveness of the visual language it is given anyway: For functions, security parameters, attack indicators, and unwanted event indicators, libraries need to be provided that the security engineer can choose from, as fits his system under consideration. The definition of these libraries is subject of the authors' current research. Regarding the visual language, the next steps are **(1)** to completely design the visual language along the five pillars found in this paper, **(2)** to create a software demonstrator and an electronic data model for efficiently working with the visual language as demanded in pillar five, **(3)** to validate the effectiveness of the visual language in case studies, using the software demonstrator in real-life engineering projects at a chemical producer and an automation component manufacturer, and **(4)** to work towards standardizing the electronic data model that undergirds the visual language in UML, AutomationML, and as a submodel of the asset administration shell (AAS).

When using the visual language as motivated in this paper in combination with libraries, engineers designing an ICS do not need to bring anything more to the security engineering table than the engineering knowledge they possess anyway. This knowledge is all they need for making informed security decisions and documenting them so they're still traceable later, during operations of the system. The visual language helps them to quickly grasp the security-relevant information and guides them through the process of making the security decisions to design a defensible ICS.

REFERENCES

[1] P. Kieseberg and E. Weippl, "Security challenges in cyber-physical production systems," in *Software Quality: Methods and Tools for Better Software and Systems*, D. Winkler, S. Biffl, and J. Bergsmann, Eds. Cham, Switzerland: Springer, 2018, pp. 3–16.

[2] M. Eckhart, A. Ekelhart, A. Luder, S. Biffl, and E. Weippl, "Security development lifecycle for cyber-physical production systems," in *Proc. IECON 45th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2019, pp. 3004–3011.

[3] J. F. Ruiz, A. Maña, and C. Rudolph, "An integrated security and systems engineering process and modelling framework," *Comput. J.*, vol. 58, no. 10, pp. 2328–2350, Oct. 2015, doi: 10.1093/comjnl/bxu152.

[4] M. Glawe, C. Tebbe, A. Fay, and K.-H. Niemann, "Knowledge-based engineering of automation systems using ontologies and engineering data," in *Proc. 7th Int. Joint Conf. Knowl. Discovery, Knowl. Eng. Knowl. Manage.*, 2015, pp. 291–300.

[5] *Engineering and Execution of PCT Projects in Process Industry*, document NA35, NAMUR e. V., 2019.

[6] M. Hollender, *Collaborative Process Automation Systems*. Durham, NC, USA: Research Triangle Park, 2010.

[7] L. Lemaire, J. Lapon, B. D. Decker, and V. Naessens, "A SysML extension for security analysis of industrial control systems," in *Proc. Electron. Workshops Comput.*, Sep. 2014, pp. 1–9. [Online]. Available: http://ewic.bcs.org/content/ConWebDoc/53223

[8] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke, "Model-based security metrics using ADversary VIew security evaluation (ADVISE)," in *Proc. 8th Int. Conf. Quant. Eval. Syst.*, Sep. 2011, pp. 191–200. [Online]. Available: http://ieeexplore.ieee.org/document/6042046/

[9] T. Sommestad, M. Ekstedt, and H. Holm, "The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures," *IEEE Syst. J.*, vol. 7, no. 3, pp. 363–373, Sep. 2013, doi: 10.1109/JSYST.2012.2221853.

[10] A. H. Vu, N. O. Tippenhauer, B. Chen, D. M. Nicol, and Z. Kalbar-Czyk, "CyberSAGE: A tool for automatic security assessment of cyber-physical systems," in *Quantitative Evaluation of Systems*, G. Norman and W. Sanders, Eds. Cham, Switzerland: Springer, 2014, pp. 384–387.

[11] M. Eckhart, A. Ekelhart, and E. Weippl, "Automated security risk identification using AutomationML-based engineering data," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 3, pp. 1655–1672, May 2022, doi: 10.1109/TDSC.2020.3033150.

[12] P. Dedousis, G. Stergiopoulos, G. Arampatzis, and D. Gritzalis, "Towards integrating security in industrial engineering design practices," in *Proc. 18th Int. Conf. Secur. Cryptogr.*, 2021, pp. 161–172.

[13] (2022). *Department of Homeland Security, USA, Cyber Security Evaluation Tool (CSET)*. Accessed: Mar. 20, 2022. [Online]. Available: https://github.com/cisagov/cset/releases

[14] S. Fluchs, R. Drath, and A. Fay, "A security decision base: How to prepare security by design decisions for industrial control systems," in *Proc. 17th EKA (Fachtagung Entwurf Komplexer Automatisierungssysteme)*, Magdeburg, Germany, 2022, pp. 1–24.

[15] S. Fluchs, E. Tasten, M. Mertens, A. Horch, R. Drath, and A. Fay, "Security by design integration mechanisms for industrial control systems," in *Proc. IECON 48th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2022, pp. 1–6.

[16] J. Bertin, *Semiology of Graphics*. Madison, WI, USA: Univ. Wisconsin Press, 1983.

[17] S. M. Kosslyn, "Graphics and human information processing: A review of five books," *J. Amer. Stat. Assoc.*, vol. 80, no. 391, pp. 499–512, Sep. 1985, doi: 10.1080/01621459.1985.10478147.

[18] J. H. Larkin and H. A. Simon, "Why a diagram is (sometimes) worth ten thousand words," *Cognit. Sci.*, vol. 11, no. 1, pp. 65–100, Jan. 1987, doi: 10.1111/j.1551-6708.1987.tb00863.x.

[19] P. Goolkasian, "Pictures, words, and sounds: From which format are we best able to reason?" *J. Gen. Psychol.*, vol. 127, no. 4, pp. 439–459, Oct. 2000, doi: 10.1080/00221300009598596.

[20] D. Moody, "The 'physics' of notations: Toward a scientific basis for constructing visual notations in software engineering," *IEEE Trans. Softw. Eng.*, vol. 35, no. 6, pp. 756–779, Nov./Dec. 2009, doi: 10.1109/TSE.2009.67.

[21] D. Kahneman, *Thinking, Fast and Slow*, 22nd ed. New York, NY, USA: Farrar, Straus and Giroux, 2011.

[22] D. Moody, "What makes a good diagram? Improving the cognitive effectiveness of diagrams in IS development," in *Advances in Information Systems Development*, W. Wojtkowski, W. G. Wojtkowski, J. Zupan-Cic, G. Magyar, and G. Knapp, Eds. Boston, MA, USA: Springer, 2007, pp. 481–492, doi: 10.1007/978-0-387-70802-7_40.

[23] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin, and T. M. Shortell, *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, 4th ed. Hoboken, NJ, USA: Wiley, 2015.

[24] U. Frank, "Domain-specific modeling languages: Requirements analysis and design guidelines," in *Domain Engineering*. Cham, Switzerland: Springer, 2013, pp. 133–157.

[25] S. Krug, *Don't Make Me Think, Revisited: A Common Sense Ap-Proach to Web Usability*, 3rd ed. Greenwich, CT, USA: New Riders, 2013.

[26] G. Rathwell, "PERA control and information systems lead engineer's guide industry sector: Oil & gas," Purdue Enterprise Reference Archit. (PERA), Tech. Rep., 2018. [Online]. Available: https://www.pera.net

[27] *Diagrams for the Chemical and Petrochemical Industry—Part 1: Specification of Diagrams*, Standard ISO 10628-1, ISO, 2014.

[28] *Diagrams for the Chemical and Petrochemical Industry—Part 2: Graphical Symbols*, Standard ISO 10628-2, ISO, 2012.

[29] *Representation of Process Control Engineering Requests in P&I Diagrams and Data Exchange Between P&ID Tools and PCE-CAE Tools*, Standard IEC 62424, IEC, 2016.

[30] *Programmable Controllers—Part 3: Pro-Gramming Languages*, Standard IEC 61131-3, IEC, 2013.

[31] H. Holm, M. Ekstedt, T. Sommestad, and L. NordstrM, "CySeMoL: A tool for cyber security analysis of enterprises," in *Proc. 22nd Int. Conf. Exhib. Electr. Distrib. (CIRED)*, 2013, p. 1109.

[32] C. B. Haley, R. Laney, J. D. Moffett, and B. Nuseibeh, "Security requirements engineering: A framework for representation and analysis," *IEEE Trans. Softw. Eng.*, vol. 34, no. 1, pp. 133–153, Jan. 2008, doi: 10.1109/TSE.2007.70754.

[33] D. Hatebur, M. Heisel, and H. Schmidt, "A pattern system for security requirements engineering," in *Proc. 2nd Int. Conf. Availability, Rel. Secur. (ARES)*, 2007, pp. 356–365. [Online]. Available: http://ieeexplore.ieee.org/document/4159824/

[34] Object Management Group (OMG). *Unified Modeling Language (UML) Specification v2.5.1*. Accessed: Aug. 30, 2022. [Online]. Available: https://www.omg.org/spec/UML/2.5.1/About-UML/

[35] J. Jürjens, "UMLsec: Extending UML for secure systems development," in *Proc. UML Unified Modeling Lang.*, G. Goos, J. Hartmanis, J. van Leeuwen, J.-M. Jézéquel, H. Hussmann, S. Cook, Eds. Berlin, Germany: Springer, 2002, pp. 412–425. [Online]. Available: http://link.springer.com/10.1007/3-540-45800-X_32

[36] J. Jürjens, *Secure Systems Development With UML*. Berlin, Germany: Springer, 2005.

[37] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-based modeling language for model-driven security," in *Proc. UML Unified Modeling Lang.*, G. Goos, J. Hartmanis, J. van Leeuwen, J.-M. Jézéquel, H. Hussmann, S. Cook, Eds. Berlin, Germany: Springer, 2002, pp. 426–441, doi: 10.1007/3-540-45800-X_33.

[38] J. McDermott and C. Fox, "Using abuse case models for security requirements analysis," in *Proc. 15th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, 2022, pp. 55–64. [Online]. Available: http://ieeexplore.ieee.org/document/816013/

[39] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Eng.*, vol. 10, pp. 34–44, Jan. 2005, doi: 10.1007/s00766-004-0194-4.

[40] G. Popp, J. Jurjens, G. Wimmel, and R. Breu, "Security-critical system development with extended use cases," in *Proc. 10th Asia–Pacific Softw. Eng. Conf.*, 2022, pp. 478–487. [Online]. Available: http://ieeexplore.ieee.org/document/1254403/

[41] M. Vasilevskaya, *Designing Security-Enhanced Embedded Systems: Bridging Two Islands of Expertise: Linköping*. Cary, NC, USA: Univ. Electronic Press, 2013. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-98213

[42] D. Mellado, E. Fernández-Medina, and M. Piattini, "Applying a security requirements engineering process," in *Computer Security ESORICS 2006*, D. Hutchison, Ed. Berlin, Germany: Springer, 2006, pp. 192–206, doi: 10.1007/11863908_13.

[43] J. L. Vivas, J. A. Montenegro, and J. López, "Towards a business process-driven framework for security engineering with the UML," in *Information Security*, G. Goos, J. Hartmanis, J. van Leeuwen, C. Boyd, and W. Mao, Eds. Berlin, Germany: Springer, 2003, pp. 381–395, doi: 10.1007/10958513_29.

[44] *Object Management Group (OMG), System Modeling Language (SysML) Specification v1.6*. Accessed: Aug. 31, 2022. [Online]. Available: https://www.omg.org/spec/SysML/1.6/About-SysML/

[45] E. Crawley, B. Cameron, and D. Selva, *System Architecture: Strategy and Product Development for Complex Systems*. Edinburgh, Scotland: Pearson, 2016.

[46] L. Apvrille and Y. Roudier, "SysML-Sec: A SysML environment for the design and de-velopment of secure embedded systems," in *Proc. APCOSEC, Asia–Pacific Council Syst. Eng.*, 2013, pp. 8–11.

[47] D. Mažeika and R. Butleris, "MBSEsec: Model-based systems engineering method for creating secure systems," *Appl. Sci.*, vol. 10, no. 7, p. 2574, Apr. 2020, doi: 10.3390/app10072574.

[48] E. S. Yu, "Social modeling and I," in *Conceptual Modeling: Foundations and Applications*, A. T. Borgida, V. K. Chaudhri, P. Giorgini, and E. S. Yu, Eds. Berlin, Germany: Springer, 2009, pp. 99–121, doi: 10.1007/978-3-642-02463-4_7.

[49] J. Mylopoulos and J. Castro, "Tropos: A framework for requirements-driven software development," in *Information Systems Engineering: State of the Art and Research Themes* (Lecture Notes in Computer Science). Springer, 2000, pp. 261–273.

[50] H. Mouratidis and P. Giorgini, "Secure Tropos: A security-oriented extension of the tropos methodology," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 17, no. 2, pp. 285–309, Apr. 2007, doi: 10.1142/S0218194007003240.

[51] Y. Asnar, P. Giorgini, F. Massacci, and N. Zannone, "From trust to dependability through risk analysis," in *Proc. 2nd Int. Conf. Availability, Rel. Secur. (ARES)*, 2007, pp. 19–26.

[52] N. Mayer, A. Rifaut, and E. Dubois, "Towards a risk-based security requirements engineering framework," in *Proc. REFSQ*, 2005, pp. 1–15.

[53] H. Mouratidis, N. Argyropoulos, and S. Shei, "Security requirements engineering for cloud computing: The secure Tropos approach," in *Domain-Specific Conceptual Modeling*, D. Karagiannis, H. C. Mayr, J. Mylopoulos, Eds. Cham, Switzerland: Springer, 2016, pp. 357–380, doi: 10.1007/978-3-319-39417-6_16.

[54] F. Massacci and N. Zannone, "Detecting conflicts between functional and security requirements with secure Tropos: John Rusnak and the allied Irish bank," in *Social Modeling for Requirements Engineering*. Cambridge, MA, USA: MIT Press, 2008.

[55] G. Elahi and E. Yu, "A goal oriented approach for modeling and analyzing security trade-offs," in *Conceptual Modeling ER 2007*, C. Parent, K.-D. Schewe, V. C. Storey, and B. Thalheim, Eds. Berlin, Germany: Springer, 2007, pp. 375–390, doi: 10.1007/978-3-540-75563-0_26.

[56] M. S. Lund, B. Solhaug, and K. Stølen, *Model-Driven Risk Analysis*. Berlin, Germany: Springer, 2011, doi: 10.1007/978-3-642-12323-8.

[57] M. Almorsy and J. Grundy, "SecDSVL: A domain-specific isual language to sup-port enterprise security modelling," in *Proc. 23rd Austral. Softw. Eng. Conf.*, 2014, pp. 152–161. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6824120

[58] S. Geraldo and R. Woods. (Dec. 16, 2020). *Virtual Expo: CISA's Cybersecurity Evaluation Tool (CSET)*. [Online]. Available: https://www.cisa.gov/sites/default/files/publications/2020-seminars-cset-508.pdf

[59] *Security for Industrial Automation and Control Systems—Part 3–2: Security Risk Assessment for System Design*, Standard IEC 62443-3-2, IEC, 2020.

[60] S. Lipner and M. Howard, "The security development lifecycle," in *SDL: A Process for Developing Demonstrably More Secure Software*. USA: Microsoft Press, 2006.

[61] K. A. Kozar. *The Technique of Data Flow Diagramming*. Accessed: Aug. 31, 2022. [Online]. Available: https://spot.colorado.edu/~kozar/DFDtechnique.html

[62] T. Drewry. *Data Flow Diagrams*. [Online]. Available: http://www.cems.uwe.ac.uk/~kg-doyle/tdrewry/dfds.htm

[63] B. Schneier, "Attack trees," *Dr. Dobb's J.*, vol. 24, no. 12, pp. 21–29, 1999. [Online]. Available: https://www.schneier.com/academic/archives/1999/12/attack_trees.html

[64] E. J. Byres, M. Franz, and D. Miller, "The use of attack trees in assessing vulnerabilities in SCADA systems," in *Proc. Int. Infrastruct. Survivability Workshop*, 2004, p. 9.

[65] J. Hoff, "Creating attack graphs for adversary emulation, simulation and purple teaming in industrial control system (ICS) environments," M.S. thesis, Dept. Math. Inform., FernUniversität, Hagen, Germany, 2021. [Online]. Available: https://pull-the-plug.net/thesis/

[66] *MITRE, ATT&CK for ICS*. Accessed: May 29 2022. [Online]. Available: https://attack.mitre.org/matrices/ics/

[67] A. van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," in *Proc. 26th Int. Conf. Softw. Eng.*, 2022, pp. 148–157. [Online]. Available: http://ieeexplore.ieee.org/document/1317437/

[68] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer, "Foundations of attack-defense trees," in *Formal Aspects of Security and Trust*, P. Degano, S. Etalle, and J. Guttman, Eds. Berlin, Germany: Springer, 2011, pp. 80–95. [Online]. Available: https://www.researchgate.net/publication/221026894_Foundations_of_Attack-Defense_Trees

[69] D. Moody and J. van Hillegersberg, "Evaluating the visual syntax of UML: An analysis of the cognitive effectiveness of the UML family of diagrams," in *Software Language Engineering*, D. Gašević, R. Lämmel, and E. van Wyk, Eds. Berlin, Germany: Springer, 2009, pp. 16–34.

[70] S. Fluchs and H. Rudolph, "Making OT security engineering deserve its name: A guide to security engineering for OT engineers," 2019. Accessed: Sep. 1, 2022. [Online]. Available: https://www.controlglobal.com/articles/2019/making-ot-security-engineering-deserve-its-name

[71] A. A. Bochman and S. G. Freeman, *Countering Cyber Sabotage. Introducing Consequence-Driven, Cyber-Informed Engineering (CCE)*, 1st ed. Boca Raton, FL, USA: CRC Press, 2021.

[72] C. Raspotnig, V. Katta, P. Karpati, and A. L. Opdahl, "Enhancing CHASSIS: A method for combining safety and security," in *Proc. Int. Conf. Availability, Rel. Secur.* Regensburg, Germany: IEEE, 2013, pp. 766–773, doi: 10.1109/ARES.2013.102.

SARAH FLUCHS was born in 1990. She received the B.Sc. degree in mechanical engineering and the M.Sc. degree in automation engineering from RWTH Aachen University, Germany, in 2015 and 2017, respectively. She is currently pursuing the Ph.D. degree with Helmut Schmidt University, Hamburg, Germany, supervised by Prof. Fay and Prof. Drath.

After written her master's thesis at the German Federal Ministry for Information Security (BSI), in 2017, she joined as a Security Consultant at admeritia, Langenfeld, Germany, an OT security consulting company. She was the Head of Security Engineering at admeritia, from 2018 to 2020 and the CTO, since 2020. Her main research interests include security engineering for industrial control and automation systems and critical infrastructures, security by design as part of a broader resilience engineering approach, and the use of human-readable diagrams and machine-readable models for security engineering.

Ms. Fluchs is a member of the International Society of Automation (ISA), where she has been the Director of the Standards and Practices Board, since 2021, and an Expert at the International Electrotechnical Commission (IEC).

RAINER DRATH was born in 1970. He received the Diploma and Ph.D. degrees in automation engineering from the Technical University of Ilmenau, Germany, in 1995 and 1999, respectively.

He worked for 17 years at the German ABB Corporate Research Center. In 2017, he was appointed as a Professor of mechatronic systems engineering at Pforzheim University. He is the author of several reference books, coauthor of several IEC standards, and Architect of AutomationML. His research interests include methods of improving automation engineering in factory and process automation, especially in the area of data models and data integration along the engineering tool chains.

Prof. Drath was a Board Member of the AutomationML Society. He received multiple IEEE best paper awards.

ALEXANDER FAY (Senior Member, IEEE) was born in 1970. He received the Diploma and Ph.D. degrees in electrical engineering from the Technical University of Braunschweig, Braunschweig, Germany, in 1995 and 1999, respectively.

He had worked for five years at the ABB Corporate Research Laboratories, Heidelberg and Ladenburg. Before, he was appointed as a Full Professor at the Institute of Automation Technology, Helmut Schmidt University, Hamburg, Germany, in 2004. His main research interests include models and methods for the engineering of large automated systems, especially in the process and manufacturing industries, and in buildings and transportation systems.

Prof. Fay has served as a member for the AdCom of the IEEE Industrial Electronics Society. He was the Program Co-Chair of the IEEE International Conference on Automation Science and Engineering (CASE), in 2018. From 2011 to 2017, he has served as an Associate Editor for the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.