Structured Semantics for the CORAS Security Risk Modelling Language

Heidi E. I. Dahl[1,4], Ida Hogganvik[2,3] and Ketil Stølen[1,3]

[1] SINTEF ICT, Oslo, Norway
[2] Scandpower Risk Management AS, Kjeller, Norway
[3] Department of Informatics, UiO, Oslo, Norway
heidi.dahl@sintef.no, iho@scandpower.com, ketil.stolen@sintef.no

Abstract. The CORAS security risk modelling language is a customised graphical language for communication, documentation and analysis of security threat and risk scenarios. This paper presents a semantics for the CORAS language. The semantics is structured in that it provides step-by-step instructions on how to correctly interpret an arbitrary CORAS diagram. The result is a readable paragraph of English. This enables users of the CORAS language to easily extract the intended meaning of a given diagram. The semantics is modular in the sense that the semantics of any diagram can be deduced from the semantics of its elements and relations.

1 Introduction

CORAS is a method for security risk analysis [5]. It comes with a specialised language for communication, documentation and analysis of security threat and risk scenarios. The language was originally defined as a UML [16] profile [15, 17], and has later been customised and refined in several aspects, based on experiences from industrial case studies, and by empirical investigations documented in [6], [7] and [8].

The CORAS language is in particular intended to support brainstorming sessions used to identify and estimate security risks. Such brainstorming sessions are characterised by the involvement of people with thorough knowledge of specific, but only partly overlapping aspects of the target of analysis. Typical participants are the intended users of the target, its designers, developers, and relevant decision makers. These people have normally quite different backgrounds and it may be difficult for the analysts to make them work well together as a group. Our experiences indicate that the CORAS language improves both the efficiency of the analysis process and the quality of the results.

We claim that our graphical approach to security risk modelling contributes to solving three issues related to security analysis:

- *How to facilitate communication in a group consisting of people with different backgrounds and competences:* Our aim has been to provide the participants

[4] Main author

with a means of communication that covers both technical and more high-level information, without being too complicated to understand. Offering a common basis for communication will hopefully reduce misunderstandings and thereby give a more correct risk picture.
- *How to estimate the likelihoods and consequences of identified risks:* In practice, reliable data on which this can be based is often not available. The participants must use their expert knowledge, experience and familiarity with the domain to estimate both the likelihoods and the consequences of incidents that might not have happened yet. Our aim has been to offer a structured, graphical risk picture to make the complexity more manageable. A graphical representation may illustrate who or what caused the incidents and the weaknesses in the system that made them possible.
- *How to document the security analysis in a comprehensible manner:* The findings of a security analysis constitute vital information not only to the participants in the analysis, but to the organization as a whole. Our aim has been to define a documentation method that should be more or less self-explanatory, and not rely on extensive training to be understood.

Although we have aimed at making a language that is easily understandable, situations are bound to arise where the intended meaning of a construct or an expression needs further explanation. The main contribution of this paper is the definition of a structured semantics aiming to fulfil this need. The semantics takes an arbitrary CORAS diagram and delivers its intended meaning as a readable paragraph of English. It is structured in the sense that it comes with step-by-step instructions allowing the translation to be conducted automatically. The semantics has been developed to meet the following success criteria:

1. *The translation from CORAS diagrams to English should be modular.* If we add new relations and/or elements to a diagram we have already translated, the translation of the modified diagram is the union of the translation of the original diagram with the translation of the new relations and/or elements.
2. *The resulting paragraph should be understandable English.* The purpose of the translation is to provide a description, in English, of a CORAS diagram, in order to communicate the meaning of the diagram to those not familiar with the intended meaning of the various elements and relations of the CORAS language.
3. *The translation should be easy to perform.* Anyone, even someone unacquainted with CORAS diagrams, should be able to translate a CORAS diagram into English.
4. *The translation should be possible to automate.* Automatic translation is a feature that will be implemented in the CORAS tool in the future (see `http://coras.sourceforge.net` for downloads and documentation).
5. *It should be possible to translate inconsistent diagrams, and the translation should enable the user to identify inconsistencies.* Inconsistent diagrams should still be possible to translate, and the resulting paragraph in English should be sufficiently clear to allow the user to identify the cause of the inconsistency.

The remainder of the paper is structured into four sections. Sec. 2 introduces the CORAS language. Sec. 3 provides an overview of the structured semantics and relevant notation. The semantics is divided into two main steps: the translation from the graphical to the textual syntax, which is described in Sec. 3.1, and the translation from the textual syntax to English, which is described in Sec. 3.2. Sec. 4 gives an example of the translation of a diagram. Finally, Sec. 5 presents our conclusions and related work.

2 The CORAS Language

The CORAS language originates from a UML profile developed as a part of the EU funded research project CORAS (IST-2000-25031) [1] (`http://coras.sourceforge.net`). As a result of our work to satisfy the modelling needs in a security risk analysis, the language and its guidelines have evolved into a more specialized and refined approach. The language is meant to support the analyst during the security risk analysis, and serves different purposes in each phase of the analysis. A security risk analysis is normally structured into five phases: (1) context establishment, (2) risk identification, (3) risk estimation, (4) risk evaluation and (5) treatment identification [3].

In the context establishment we employ *assets overview diagrams* to specify the parties of the security analysis and their assets. The purpose is to obtain a precise definition of what the valuable aspects of the target of analysis are, and which are the most important. From empirical investigations [6] and field trials we know that asset identification and valuation is very difficult, and that mistakes or inaccuracies made there may jeopardize the value of the whole security analysis.

During risk identification we use *threat diagrams* to identify and document how vulnerabilities may be exploited by threats to initiate unwanted incidents, and which assets they affect. The threat diagrams give a clear and easily understandable overview of the risk picture and make it easier to see who or what the threat is, how the threat works (threat scenarios) and which vulnerabilities and assets they involve.

The threat diagrams are used as input for the risk estimation phase, where unwanted incidents are assigned likelihood estimates and possible consequences. The likelihood estimation is often a difficult task, but illustrating the unwanted incidents in the correct context has proved very helpful in practice.

After the risk estimation, the magnitude of each risk can be calculated on the basis of its likelihood and consequence, and modelled in *risk overview diagrams*. The risk overview diagrams specify which threats initiate the different risks, and exactly which assets they may harm. This risk representation is then compared to predefined risk tolerance levels to decide which ones need treatments.

In the treatment identification, the threat diagrams containing the risks that cannot be tolerated are used as basis for treatment identification. In this phase the appropriate treatments are identified and modelled in *treatment diagrams*.

The resulting treatment diagrams can be seen as a plan for how to deal with the identified risks.

Communicating the results of an analysis in such a way that they are well understood by decision makers can be challenging. The CORAS language supports this by offering *treatment overview diagrams*. Treatment overview diagrams may for example be used to provide a high level summary when presenting the main findings from an analysis.

To summarise, the CORAS language consists of five different kinds of diagrams: assets overview diagrams, threat diagrams, risk overview diagrams, treatment diagrams and treatment overview diagrams. Their basic building blocks are presented in Fig. 1.

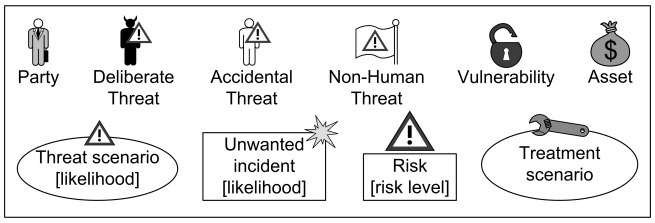

Fig. 1. Basic building blocks of the CORAS diagrams

In the rest of the paper, we focus on assets overview diagrams and threat diagrams. The semantics is defined accordingly for risk overview, treatment and treatment overview diagrams as explained in the full report [4].

2.1 Constructing an Assets Overview Diagram

Fig. 2 presents the syntax of an assets overview diagram.

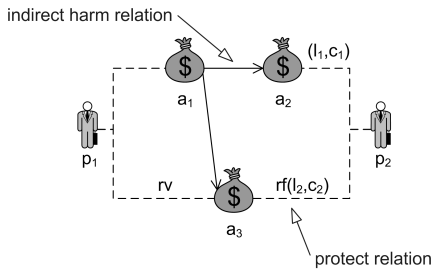

Fig. 2. Graphical syntax of assets overview diagrams

Assets overview diagrams are used early in the analysis to fix its scope. The relevant *assets* are placed in the diagram, and when appropriate connected by

indirect harm relations to indicate that harm to one asset may affect another. *Parties* may be added, and connected to assets with *protect* relations. A protect relation may be annotated with a risk level, indicating the level of risk a party is willing to accept with regards to the asset in question. The different kinds of risk levels is shown in Fig. 2: it is either a *risk value* which is either a numerical value or a linguistic term such as "low"/"medium"/"high", a *likelihood and consequence pair* giving the maximal acceptable values, or a *risk function* of such a pair. Hence, the parties are the customers, institutions or organisations on behalf of whom the analysis is carried out. In practice there is often only one party.

To summarise, assets overview diagrams are constructed from two basic building blocks, using two relations:

Basic building blocks: Asset, Party.
Relations: Protect (may be annotated with a risk level), Indirect harm.

2.2 Constructing a Threat Diagram

Fig. 3 presents the syntax of a threat diagram.

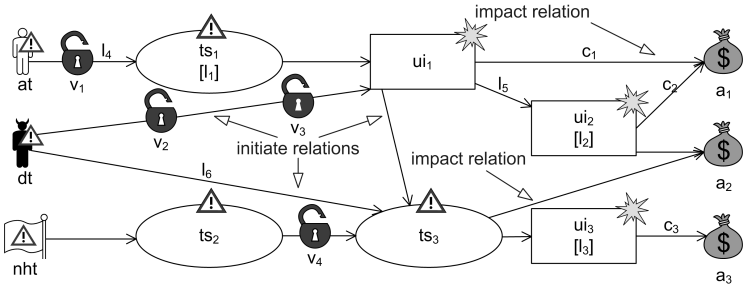

Fig. 3. Graphical syntax of threat diagrams

When constructing a threat diagram, we start by placing the *assets* to the far right, and potential *threats* to the far left. The construction of the diagram is an iterative process, and we may add more threats later on in the analysis. The assets were fixed when the assets overview diagram was constructed.

Next we place *unwanted incidents* to the left of the assets. They represent events which may have a negative impact on one or more of the assets. This *impact* relation is represented by drawing an arrow from the unwanted incident to the relevant asset, and may be annotated with a consequence value.

The next step consists in determining the different ways a threat may initiate an unwanted incident. We do this by placing *threat scenarios*, each describing a series of events, between the threats and unwanted incidents and connecting them all with *initiate* relations. An initiate relation may originate from either threats, threat scenarios or unwanted incidents, and terminate at threat scenarios

or unwanted incidents, and displays the causal relationship between the elements. In the case where a vulnerability is exploited when passing from one element to another, the vulnerability is positioned on the arrow between them.

There is also the possibility of an impact relation from a threat scenario to an asset, however this relation is mainly used in the early stages of the analysis and may not be annotated with a consequence. This has to do with the difference between unwanted incidents and threat scenarios: unwanted incidents are used to give a description of single events *that may have a consequence for an asset.* Threat scenarios are used to describe the sequences of events leading up to an unwanted incident. In the case where a threat scenario has direct consequences for an asset, a new unwanted incident should eventually be inserted to express this.

At this point, likelihoods may be added to threat scenarios, unwanted incidents and initiate relations. The likelihoods of the first two are the likelihood that they will happen at all. The likelihood of an initiate relation is the likelihood of the second element, given the first.

To summarise, threat diagrams are constructed from seven basic building blocks, using two relations:

> *Basic building blocks:* Deliberate, Accidental, and Non-Human Threat, Vulnerability, Threat Scenario, Unwanted Incident, Asset.
> *Relations:* Initiate (may be annotated with a likelihood), Impact (may in some cases be annotated with a consequence).

3 The Structured Semantics

The structured semantics for the CORAS language is divided into two separate steps:

(A) The translation of a diagram into its textual syntax, and
(B) The translation of its textual syntax into its meaning as a paragraph in English.

Hence, the semantics enables the user of CORAS to extract the meaning of an arbitrary CORAS diagram by applying first (A), then (B) (this is written as ($B \circ A$)).

Both these steps, and therefore the structured semantics, are modular: a diagram is translated relation by relation. Step (A) is described in Sec. 3.1, and Step (B) in Sec. 3.2. In both sections, we make use of the naming conventions in Table 1. For simplicity we use a (possibly decorated) p to represent a party, and a (possibly decorated) a to represent an asset, etc.

3.1 Step (A): From the Graphical to the Textual Syntax

The textual syntax of the CORAS language is defined using a standardised EBNF notation [11]. For the complete syntax, and translation rules for all the

Element	Instance
party	p
asset	a
deliberate threat	dt
accidental threat	at
non-human threat	nht
vulnerability	$v = \{v\}$
vulnerability set	$V = \{v_1, \ldots, v_n\}$

Element	Instance
threat scenario	ts
unwanted incident	ui
likelihood	l
consequence	c
risk	r
risk value	rv
risk function	rf
treatment scenario	trs

Table 1. Naming conventions

relations in the CORAS language, see the full report [4]. In this section, we explain how a diagram is translated from the graphical to the textual syntax, using the assets overview diagram as an example. The other kinds of diagrams are translated accordingly.

The EBNF grammar for the assets overview diagram is the following:

$$
\begin{aligned}
\mathit{relation} &= \mathit{protect} \mid \mathit{indirect\ harm};\\
\mathit{protect} &= \mathit{party} \overset{[\mathit{risk\ level}]}{\cdots} \mathit{asset};\\
\mathit{indirect\ harm} &= \mathit{asset} \rightarrow \mathit{asset};\\
\mathit{party} &= \mathit{identifier};\\
\mathit{asset} &= \mathit{identifier};\\
\mathit{risk\ level} &= \mathit{risk\ value} \mid \mathit{risk\ function}(\mathit{likelihood}, \mathit{consequence}) \mid\\
&\quad (\mathit{likelihood}, \mathit{consequence});\\
\mathit{risk\ value} &= \mathit{linguistic\ term} \mid \mathit{numerical\ value};\\
\mathit{likelihood} &= \mathit{linguistic\ term} \mid \mathit{numerical\ value};\\
\mathit{consequence} &= \mathit{linguistic\ term} \mid \mathit{numerical\ value};
\end{aligned}
$$

The EBNF defines the structure of the diagram, as it was explained in Sec. 2. The assets overview diagram has two relations which we want to translate: the protect relation in Fig. 4(a) and the indirect harm relation in Fig. 4(b).

(a) (b)

Fig. 4. Relations of the assets overview diagram

The translation from the graphical to the textual syntax is essentially replacing all the icons with their textual label. In the assets overview diagram, this means that the protect relation in Fig. 4(a) is translated into $p \overset{rl}{\cdots} a$, and the indirect harm relation in Fig. 4(b) into $a_1 \rightarrow a_2$.

The other diagrams are translated in the same manner.

3.2 Step (B): From the Textual Syntax to English

In this step of the structured semantics we apply the semantic function $[\![\_]\!]$ to the textual expressions resulting from Step (A), obtaining a sentence in English for each expression. We start by defining the semantics for the basic building blocks, and these definitions are then used to define the semantics for the relations.

The translation rules of the initiate and treat relations involving unwanted incidents are identical to those involving threat scenarios. The rules for the former can be obtained by replacing *ts* with *ui* in the latter.

We simplify accordingly for the three different kinds of threats, specifying the rules with *dt* for direct threat in the semantics of the initiate and treat relations. This can be replaced by either *at* or *nht* for accidental and non-human threats.

In the semantics of risks and of the protect relation, we present the rules with *rv* for risk level. This can be replaced by either a risk function $rf(l, c)$ or a likelihood and consequence pair (l, c).

The translation rules of the treat relations does not depend on the treatment category. We therefore present only the rules with *av*, the rules for the other treatment categories can be obtained by replacing *av* with *dl*, *dc*, *sh* or *re* (see the semantics of the treatment categories for definitions).

Translating the Basic Building Blocks

$$
\begin{aligned}
[\![p]\!] &:= \text{party } `p' \\
[\![a]\!] &:= \text{asset } `a' \\
[\![dt]\!] &:= \text{deliberate threat } `dt' \\
[\![at]\!] &:= \text{accidental threat } `at' \\
[\![nht]\!] &:= \text{non-human threat } `nht' \\
[\![v]\!] &:= \text{vulnerability } `v' \\
[\![V]\!] &:= \text{vulnerability set } `v_1', \ \ldots \ , `v_n' \\
[\![ts]\!] &:= \text{threat scenario } `ts' \\
[\![ts(l)]\!] &:= \text{threat scenario } `ts', \text{ which has } [\![l]\!], \\
[\![ui]\!] &:= \text{unwanted incident } `ui' \\
[\![ui(l)]\!] &:= \text{unwanted incident } `ui', \text{ which has } [\![l]\!], \\
[\![r]\!] &:= \text{risk } `r' \\
[\![r(rv)]\!] &:= \text{risk } `r', \text{ which has } [\![rv]\!],
\end{aligned}
$$

$$
\begin{aligned}
[\![\, r(rf(l,c)) \,]\!] &:= \text{risk `}r\text{', which has } [\![\, rf(l,c) \,]\!], \\
[\![\, r(l,c) \,]\!] &:= \text{risk `}r\text{', which has } [\![\, (l,c) \,]\!], \\
[\![\, trs \,]\!] &:= \text{treatment scenario `}trs\text{'} \\
[\![\, rv \,]\!] &:= \text{risk value `}rv\text{'} \\
[\![\, rf(l,c) \,]\!] &:= \text{risk function `}rf\text{' of } [\![\, (l,c) \,]\!] \\
[\![\, (l,c) \,]\!] &:= [\![\, l \,]\!] \text{ and } [\![\, c \,]\!] \\
[\![\, l \,]\!] &:= \text{likelihood `}l\text{'} \\
[\![\, c \,]\!] &:= \text{consequence `}c\text{'}
\end{aligned}
$$

Translating the Protect relation

$$[\![\, p \overset{rv}{\cdots} a \,]\!] := [\![\, p \,]\!] \text{ wants to protect the value of } [\![\, a \,]\!], \text{ but accepts } [\![\, rv \,]\!] \text{ or less}$$

Translating the Indirect harm relation

$$[\![\, a_1 \rightarrow a_2 \,]\!] := [\![\, a_2 \,]\!] \text{ may be harmed indirectly via } [\![\, a_1 \,]\!]$$

Translating the Initiate relation

$$
\begin{aligned}
[\![\, dt \xrightarrow{V_n \;\; l_1} ts(l_2) \,]\!] &:= \text{there is a } [\![\, l_1 \,]\!] \text{ that } [\![\, dt \,]\!] \text{ will exploit } [\![\, V_n \,]\!] \text{ to} \\
&\qquad \text{initiate } [\![\, ts(l_2) \,]\!] \\
[\![\, ts_1(l_1) \xrightarrow{V_n \;\; l_3} ts_2(l_2) \,]\!] &:= \text{after } [\![\, ts_1(l_1) \,]\!] \text{ has taken place, there is a } [\![\, l_3 \,]\!] \\
&\qquad \text{that } [\![\, V_n \,]\!] \text{ will be exploited to initiate } [\![\, ts_2(l_2) \,]\!] \\
[\![\, dt \rightarrow r(rv) \,]\!] &:= [\![\, dt \,]\!] \text{ may initiate } [\![\, r(rv) \,]\!] \\
[\![\, ts(l) \rightarrow r(rv) \,]\!] &:= [\![\, ts(l) \,]\!] \text{ may initiate } [\![\, r(rv) \,]\!] \\
[\![\, r_1(rv_1) \rightarrow r_2(rv_2) \,]\!] &:= [\![\, r_1(rv_1) \,]\!] \text{ may initiate } [\![\, r_2(rv_2) \,]\!]
\end{aligned}
$$

Translating the Impact relation

$$
\begin{aligned}
[\![\, ts(l) \rightarrow a \,]\!] &:= [\![\, ts(l) \,]\!] \text{ may impact } [\![\, a \,]\!] \\
[\![\, ui(l) \xrightarrow{c} a \,]\!] &:= [\![\, ui(l) \,]\!] \text{ may impact } [\![\, a \,]\!] \text{ with } [\![\, c \,]\!] \\
[\![\, r(rv) \rightarrow a \,]\!] &:= [\![\, r(rv) \,]\!] \text{ may impact } [\![\, a \,]\!]
\end{aligned}
$$

Translating the Treatment categories

$$
\begin{aligned}
[\![\, av \,]\!] &:= \text{avoids the risk} \\
[\![\, dl \,]\!] &:= \text{reduces the likelihood} \\
[\![\, dc \,]\!] &:= \text{reduces the consequences} \\
[\![\, sh \,]\!] &:= \text{shares the risk} \\
[\![\, re \,]\!] &:= \text{retains the risk}
\end{aligned}
$$

Translating the Treat relation

$$[\![trs \xrightarrow{av} dt]\!] := [\![trs]\!]\ [\![av]\!] \text{ of } [\![dt]\!] \text{ attacking the system}$$
$$[\![trs \xrightarrow{av} v]\!] := [\![trs]\!]\ [\![av]\!] \text{ of } [\![v]\!] \text{ being exploited}$$
$$[\![trs \xrightarrow{av} ts(l)]\!] := [\![trs]\!]\ [\![av]\!] \text{ of } [\![ts(l)]\!] \text{ being initiated}$$
$$[\![trs \xrightarrow{av} r(rv)]\!] := [\![trs]\!]\ [\![av]\!] \text{ of } [\![r(rv)]\!]$$
$$[\![trs \xrightarrow{av} a]\!] := [\![trs]\!]\ [\![av]\!] \text{ of } [\![a]\!] \text{ being harmed}$$

4 Example Translation

To illustrate how a diagram is translated we will use the threat diagram in Fig. 5

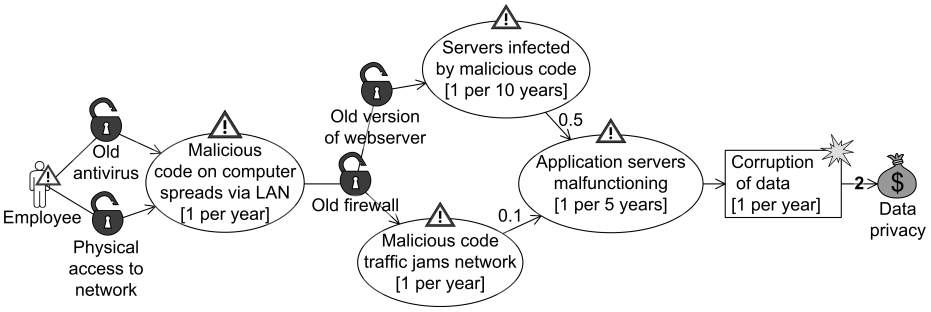

Fig. 5. Threat diagram

The threat diagram is translated relation by relation using the two-step strategy outlined above. The resulting sentences have been arranged for better readability. The diagram has 8 relations: 7 initiate relations (5 annotated with vulnerabilities and 2 with likelihoods) and 1 impact relation (annotated with a consequence). Translating top to bottom from left to right gives us:

- Accidental threat ***Employee*** may exploit vulnerability
  - ***Old antivirus*** to initiate threat scenario ***Malicious code on computer spreads via LAN***, which has likelihood ***1 per year***.
  - ***Physical access to network*** to initiate threat scenario ***Malicious code on computer spreads via LAN***, which has likelihood ***1 per year***.
- After threat scenario ***Malicious code on computer spreads via LAN***, which has likelihood ***1 per year***, has taken place,
  - vulnerability set ***Old firewall***, ***Old version of webserver*** may be exploited to initiate threat scenario ***Servers infected by malicious code***, which has likelihood ***1 per 10 years***.

  - vulnerability ***Old firewall*** may be exploited to initiate threat scenario ***Malicious code traffic jams network***, which has likelihood ***1 per year***.
- After threat scenario ***Servers infected by malicious code***, which has likelihood ***1 per 10 years***, has taken place, there is a likelihood ***0.5*** that threat scenario ***Application servers malfunctioning***, which has likelihood ***1 per 5 years***, will be initiated.
- After threat scenario ***Malicious code traffic jams network***, which has likelihood ***1 per year***, has taken place, there is a likelihood ***0.1*** that threat scenario ***Application servers malfunctioning***, which has likelihood ***1 per 5 years***, will be initiated.
- After threat scenario ***Application servers malfunctioning***, which has likelihood ***1 per 5 years***, has taken place, unwanted incident ***Corruption of data***, which has likelihood ***1 per year*** may be initiated.
- Unwanted incident ***Corruption of data***, which has likelihood ***1 per year***, may impact asset ***Data privacy*** with consequence ***2***.

We may now check whether the likelihoods of the diagram have been assigned consistently. If for example the statistically independent threat scenarios $ts_1(l_1), \ldots, ts_n(l_n)$ initiate threat scenario $ts(l)$, and the likelihoods associated with the initiate relations are $l_{i1}, \ldots, l_{in}$ respectively, then the following inequality should be true:

$$l \geq l_1 \cdot l_{i1} + \cdots + l_n \cdot l_{in} \ .$$

This is an equality only when the threat diagram is complete, i.e. when all eventualities are taken into account and all likelihoods given. If the inequality is strict, it simply means that there are causes of $ts(l)$ that are not accounted for.

In this example, all threat scenarios and unwanted incidents have been assigned likelihoods, so it is possible to check for inconsistencies with respect to the initiate relations which have also been assigned likelihoods. The relevant translations are:

- After threat scenario ***Servers infected by malicious code***, which has likelihood ***1 per 10 years***, has taken place, there is a likelihood ***0.5*** that threat scenario ***Application servers malfunctioning***, which has likelihood ***1 per 5 years***, will be initiated.
- After threat scenario ***Malicious code traffic jams network***, which has likelihood ***1 per year***, has taken place, there is a likelihood ***0.1*** that threat scenario ***Application servers malfunctioning***, which has likelihood ***1 per 5 years***, will be initiated.

The first implies that the likelihood or frequency of ***Application servers malfunctioning*** being initiated by ***Servers infected by malicious code*** is ***1 per 20 years***, and the second that the frequency of ***Application servers malfunctioning*** being initiated by ***Malicious code traffic jams network*** is ***1 per 10 years***. This tells us that the frequency of ***Application servers malfunctioning*** should be at least ***3 per 20 years*** if the diagram is to be

consistent, which is ok as the frequency is given as ***1 per 5 years*** or ***4 per 20 years***. The fact that the two frequencies are not equal tells us that if the assigned frequencies are correct, the diagram is incomplete (but consistent): there are additional causes for ***Application servers malfunctioning*** which are not accounted for.

5 Conclusion

The CORAS language has been designed to be easily understandable in order to aid communication in a security risk analysis context. Even so, situations are bound to arise where there is a need to explain the intended meaning of a construct or expression. An example of such a situation is when the analysis results are distributed to parties, within the client company, which have not been part of the analysis process.

In order to fill this need, this paper has presented a structured semantics for the CORAS security risk modelling language. We have provided instructions on how to translate the two main CORAS diagrams, via the textual syntax, into a paragraph of English.

The paper satisfies the success criteria stated at the end of Section 1 in the following sense:

1. *The translation from CORAS diagrams to English should be modular.* We divided the translation into two independent steps: (A) Graphical to textual syntax, and (B) Textual syntax to English. Both of these component translations are modular (the diagram and textual expressions are translated relation by relation) so the complete translation ($B \circ A$) is modular.
2. *The resulting paragraph should be understandable English.* The wording of the English phrases in the structured semantics is based on the descriptions used by CORAS developers to explain the diagrams to non-specialists during a CORAS security risk analysis. This gives us a translation into phrases of clear understandable English.
3. *The translation should be easy to perform.* The translation of a diagram is done by pattern matching, first by matching each relation to a translation rule and removing unwanted optional elements, then by matching the resulting textual expression to a rule in the structured semantics.
4. *The translation should be possible to automate.* The translation rules and the structured semantics are presented in such a way that the pattern matching may be done automatically. However, the structuring of the translation depends to a large degree on the structure of the original diagram. Thus it is difficult to give a general recommendation on how this is done. This means that while it is possible to automatically structure the translation to reflect the branching nature of the CORAS diagrams, a more comprehensive structuring may require human intervention unless the structure of the diagram adheres to a predefined style.

5. *It should be possible to translate inconsistent diagrams, and the translation should enable the user to identify inconsistencies.* As a CORAS diagram is translated relation by relation and not from a more global perspective, it does not matter to the translation whether or not the diagram is inconsistent. However, the inconsistencies may not be conspicuous before the translation is appropriately structured.

Related Work

Misuse cases [2, 19, 20] was an important source of inspiration in the development of the UML profile mentioned in Sec. 2. A misuse case is a kind of UML use case [12] which characterizes functionality that the system should not allow. There are a number of security oriented extensions of UML, e.g. UMLSec [13] and SecureUML [14]. These and related approaches have however all been designed to capture security properties and security aspects at a more detailed level than our language. Moreover, their focus is not on brainstorming sessions as in our case. Fault tree is a tree-notation used in fault tree analysis (FTA) [10]. The top node represents an unwanted incident, or failure, and the different events that may lead to the top event are modelled as branches of nodes, with the leaf node as the causing event. Our threat diagrams often look a bit like fault trees, but may have more than one top node. Event tree analysis (ETA) [9] focuses on illustrating the consequences of an event and the probabilities of these. Event trees can to a large extent also be simulated in our notation. Attack trees [18] aim to provide a formal and methodical way of describing the security of a system based on the attacks it may be exposed to. The notation uses a tree structure similar to fault trees, with the attack goal as the top node and different ways of achieving the goal as leaf nodes. Our approach supports this way of modelling, but facilitates in addition the specification of the attack initiators (threats) and the harm caused by the attack (damage to assets).

Further Work

The work presented in this paper is the starting point for several research activities. The most immediate would be empirical testing of the translation process and the resulting sentences. The CORAS tool will be updated to reflect the structure of the textual syntax and facilitate automatic translation.

The development of the CORAS method and language continues in several projects at SINTEF ICT, building on experiences from industrial case studies. There is also ongoing work aiming for an integrated approach to security and usability analysis.

Acknowledgements

The research for this paper has been funded by the SECURIS (152839/220) and DIGIT (180052/S10) projects of the Research Council of Norway, and the EU-project S3MS (IST-2006-027004). The authors thank Iselin Engan, Mass Soldal Lund and Atle Refsdal for valuable input.

References

[1] Jan Øyvind Aagedal, Folker den Braber, Theo Dimitrakos, Bjørn Axel Gran, Dimitris Raptis, and Ketil Stølen. Model-based risk assessment to improve enterprise security. In *EDOC'02*, pages 51–64. IEEE Computer Society, 2002.

[2] Ian F. Alexander. Misuse cases: Use cases with hostile intent. *IEEE Software*, 20(1):58–66, 2003.

[3] AS/NZS 4360:2004. *Australian/New Zealand Standard for Risk Management*, 2004.

[4] Heidi E. I. Dahl, Ida Hogganvik, and Ketil Stølen. Structured semantics for the CORAS security risk modelling language. Technical Report A970, SINTEF ICT, 2007.

[5] Folker den Braber, Ida Hogganvik, Mass Soldal Lund, Ketil Stølen, and Fredrik Vraalsen. Model-based security analysis in seven steps – a guided tour to the CORAS method. *BT Technology Journal*, 25(1):101–117, 2007.

[6] Ida Hogganvik and Ketil Stølen. On the comprehension of security risk scenarios. In *IWPC'05*, pages 115–124. IEEE Computer Society, 2005.

[7] Ida Hogganvik and Ketil Stølen. Risk Analysis Terminology for IT systems: Does it match Intuition? In *ISESE'05*, pages 13–23. IEEE Computer Society, 2005.

[8] Ida Hogganvik and Ketil Stølen. A Graphical Approach to Risk Identification, Motivated by Empirical Investigations. In *MoDELS'06*, volume 4199 of *LNCS*, pages 574–588. Springer, 2006.

[9] IEC60300. *Event Tree Analysis in Dependability management – Part 3: Application guide – Section 9: Risk analysis of technological systems.* 1995.

[10] IEC61025. *Fault Tree Analysis (FTA).* 1990.

[11] ISO/IEC 14977:1996(E). *Information technology — Syntactic metalanguage — Extended BNF*, first edition, 1996.

[12] Ivar Jacobson, Magnus Christenson, Patrik Jonsson, and Gunnar Övergaard. *Object-Oriented Software Engineering. A Use Case Driven Approach.* Addison-Wesley, 1992.

[13] Jan Jürjens. *Secure Systems Development with UML.* Springer, 2005.

[14] Torsten Lodderstedt, David A. Basin, and Jürgen Doser. SecureUML: A UML-based modeling language for model-driven security. In *UML'02*, volume 2460 of *LNCS*, pages 426–441. Springer, 2002.

[15] Mass Soldal Lund, Ida Hogganvik, Seehusen Fredrik, and Ketil Stølen. UML profile for security assessment. Technical Report STF40 A03066, SINTEF ICT, 2003.

[16] OMG. *Unified Modeling Language Specification, version 2.0*, 2004.

[17] OMG. *UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms*, 2005.

[18] Bruce Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal of Software Tools*, 24(12):21–29, December 1999.

[19] Guttorm Sindre and Andreas L. Opdahl. Eliciting security requirements with misuse cases. In *TOOLS-PACIFIC'00*, pages 120–131, 2000.

[20] Guttorm Sindre and Andreas L. Opdahl. Templates for misuse case description. In *REFSQ'01*, pages 125–136, 2001.