Using Dependent CORAS Diagrams to Analyse Mutual Dependency

Gyrd Brændeland[1,2], Heidi E. I. Dahl[1], Iselin Engan[1] and Ketil Stølen[1,2]

[1] SINTEF ICT, Oslo, Norway
[2] Department of Informatics, UiO, Oslo, Norway
gyrd.brendeland@sintef.no, heidi.dahl@sintef.no, iselin.engan@sintef.no, ketil.stolen@sintef.no

Abstract. The CORAS method for security risk analysis provides a customized language, the CORAS diagrams, for threat and risk modelling. In this paper, we extend this language to capture context dependencies, and use it as a means to analyse mutual dependency. We refer to the extension as dependent CORAS diagrams. We define a textual syntax using EBNF and explain how a dependent CORAS diagram may be schematically translated via the textual syntax into a paragraph in English, characterizing its intended meaning. Then we demonstrate the suitability of the language by means of a core example.

1 Introduction

CORAS [1] is a method for conducting security risk analysis, which is abbreviated to 'security analysis' in the rest of this paper. CORAS provides a customised language, the CORAS diagrams, for threat and risk modelling, and comes with detailed guidelines explaining how the language should be used to capture and model relevant information during the various stages of the security analysis. In this respect CORAS is model-based. The Unified Modelling Language (UML) [18] is typically used to model the target of the analysis. For documenting intermediate results and for presenting the overall conclusions we use CORAS diagrams which are inspired by UML. The CORAS method provides a computerised tool designed to support documenting, maintaining and reporting analysis results through risk modelling, table-based documentation, consistency checking and more.[3]

The main contributions of this paper are: (1) The proposal of dependent CORAS diagrams as a means to capture context dependencies, and (2) the outline of a general strategy for analysing mutual dependencies using dependent CORAS diagrams. In particular, we show how to compose the results from analysing different subcomponents when analysing a composite system.

The rest of the paper is organized as follows: Section 2 presents a subset of the CORAS language. Section 3 introduces dependent CORAS diagrams. We provide a textual syntax in EBNF [10], and a schematic translation of any

[3] The tool may be downloaded from http://coras.sourceforge.net/

dependent CORAS threat diagram into English. Section 4 presents a set of deduction rules for reasoning about dependent threat diagrams. In Section 5 we use an example to illustrate the suitability of the new features to analyse and reason about mutual dependency. Finally, in Section 6, we summarize the main results and relate our work to the existing literature.

2 CORAS Diagrams – The Basics

CORAS diagrams have been designed to document, analyse, and communicate security risk relevant information. It uses simple icons and relations between these to support the various phases of the analysis process to make diagrams that are easy to read and that are suitable as a medium for communication between stakeholders of diverse backgrounds. In particular, CORAS diagrams are meant to be used during brainstorming sessions where the discussion is documented along the way.

There are five distinct phases of security analysis according to the CORAS method: (1) context identification, (2) risk identification, (3) risk estimation, (4) risk evaluation and (5) treatment identification. Each of these phases is documented using a specific kind of CORAS diagram. The five kinds of CORAS diagrams are asset overview diagrams, threat diagrams, risk overview diagrams, treatment diagrams, and treatment overview diagrams.

In this paper, we focus on threat diagrams, which are used during the risk identification and estimation phases of the analysis. However, the presented approach to capture and analyse dependency carries over to the full CORAS language.

In the next two subsections, we present the syntax and semantics of ordinary threat diagrams as defined in [4]. In Section 3 this basic approach to threat modelling is then generalized to capture context dependency.

2.1 Syntax of Threat Diagrams

Threat diagrams describe how different threats exploit vulnerabilities to initiate threat scenarios and unwanted incidents, and how these unwanted incidents impact the assets to be protected.

The basic building blocks of threat diagrams are as follows: threats (deliberate, accidental and non-human), vulnerabilities, threat scenarios, unwanted incidents and assets. Figure 1 presents the icons representing the basic building blocks, and Figure 2 presents the syntax of a threat diagram.

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. When the threat diagrams are constructed we have typically already identified the assets of relevance and documented them in an asset overview diagram.

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. An

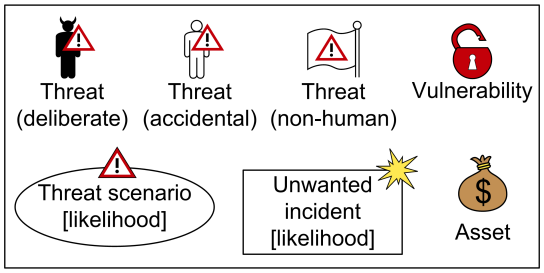

Fig. 1. Basic building blocks of CORAS threat diagrams

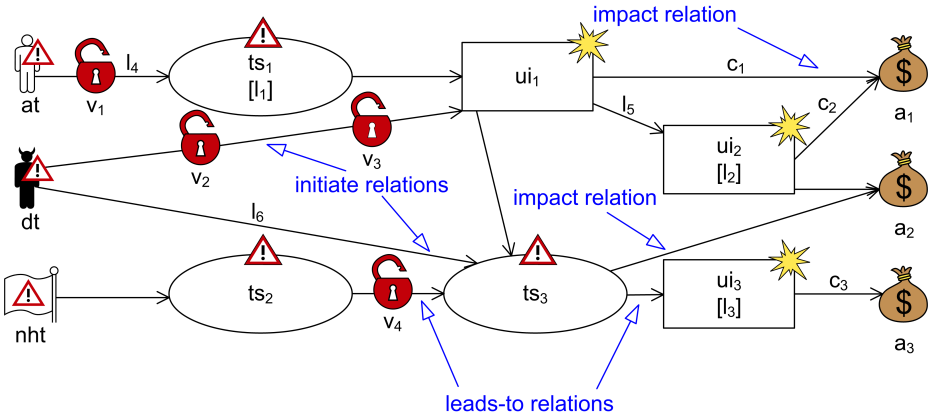

Fig. 2. Syntax of CORAS threat diagrams

impact relation is represented by an arrow from the unwanted incident to the relevant asset, and may be annotated with a consequence value (c_1, c_2 and c_3 in Figure 2).

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 and leads-to relations. An initiate relation originates in a threat and terminates in a threat scenario or an unwanted incident. Leads-to relations connect threat scenarios and unwanted incidents, and together with initiate relations display the causal relationship between the elements.

Initiate and leads-to relations, unwanted incidents and threat scenarios may all be annotated by likelihood values (such as l_3, l_4, l_5 in Figure 2). In the case where a vulnerability is exploited when passing from one element to another, the vulnerability is positioned on the arrow between them.

The graphical syntax has been carefully designed to maximize the usability of the language. Although helpful in practical modelling situations, the graphical syntax is rather cumbersome to work with when defining the semantics and rules

for the CORAS language. For this purpose we also provide an abstract textual syntax. The abstract textual syntax for threat diagrams is defined in EBNF as follows:

$$
\begin{aligned}
\mathit{diagram} &= \left(\{\mathit{vertex}\}^{-}, \{\mathit{relation}\}\right);\\
\mathit{vertex} &= \mathit{threat} \mid \mathit{threat\ scenario} \mid \mathit{unwanted\ incident} \mid \mathit{asset};\\
\mathit{relation} &= \mathit{initiate} \mid \mathit{leads\text{-}to} \mid \mathit{impact};\\
\mathit{initiate} &= \mathit{threat} \xrightarrow{[\mathit{vulnerability\ set}][\mathit{likelihood}]} \mathit{threat\ scenario} \mid\\
&\quad \mathit{threat} \xrightarrow{[\mathit{vulnerability\ set}][\mathit{likelihood}]} \mathit{unwanted\ incident};\\
\mathit{leads\text{-}to} &= \mathit{threat\ scenario} \xrightarrow{[\mathit{vulnerability\ set}][\mathit{likelihood}]} \mathit{threat\ scenario} \mid\\
&\quad \mathit{threat\ scenario} \xrightarrow{[\mathit{vulnerability\ set}][\mathit{likelihood}]} \mathit{unwanted\ incident} \mid\\
&\quad \mathit{unwanted\ incident} \xrightarrow{[\mathit{vulnerability\ set}][\mathit{likelihood}]} \mathit{threat\ scenario} \mid\\
&\quad \mathit{unwanted\ incident} \xrightarrow{[\mathit{vulnerability\ set}][\mathit{likelihood}]} \mathit{unwanted\ incident};\\
\mathit{impact} &= \mathit{unwanted\ incident} \xrightarrow{[\mathit{consequence}]} \mathit{asset} \mid\\
&\quad \mathit{threat\ scenario} \rightarrow \mathit{asset};\\
\mathit{threat} &= \mathit{deliberate\ threat} \mid \mathit{accidental\ threat} \mid \mathit{non\text{-}human\ threat};\\
\mathit{deliberate\ threat} &= \mathit{identifier};\\
\mathit{accidental\ threat} &= \mathit{identifier};\\
\mathit{non\text{-}human\ threat} &= \mathit{identifier};\\
\mathit{vulnerability\ set} &= \{\mathit{vulnerability}\}^{-};\\
\mathit{vulnerability} &= \mathit{identifier};\\
\mathit{threat\ scenario} &= \mathit{identifier}\,[(\mathit{likelihood})];\\
\mathit{unwanted\ incident} &= \mathit{identifier}\,[(\mathit{likelihood})];\\
\mathit{asset} &= \mathit{identifier};\\
\mathit{likelihood} &= \mathit{linguistic\ term} \mid \mathit{numerical\ value};\\
\mathit{consequence} &= \mathit{linguistic\ term} \mid \mathit{numerical\ value};
\end{aligned}
$$

2.2 Semantics of Threat Diagrams

The semantics of CORAS diagrams is informal in the sense that the meaning of a diagram is captured by a paragraph in structured English. By structured in this context we mean that any CORAS diagram may be schematically translated (e.g. by a computer) into a paragraph in English characterizing its intended meaning. The semantics is expressed in English to allow the meaning of diagrams to be understood also by non-technical people. This does not mean that we do not see

the value of having an additional formal semantics, but this is an issue of further work.

The CORAS semantics 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 threat diagram by applying first (A), then (B). Both these steps, and therefore the structured semantics, are modular: a diagram is translated vertex by vertex and relation by relation. For simplicity we use dt to represent a deliberate threat, a to represent an asset, etc., as outlined in Table 1.

Vertex	**Instance**
asset	a
deliberate threat	dt
accidental threat	at
non-human threat	nht
threat scenario	ts
unwanted incident	ui
Annotation	**Instance**
vulnerability	$v = \{v\} = V_1$
vulnerability set	$V_n = \{v_1, \ldots, v_n\}$
likelihood	l
consequence	c

Table 1. Naming conventions

(A) Translation from the graphical into the textual syntax
To translate a vertex from the graphical to the textual syntax, the icon is simply replaced by its label, or name. Relations are still represented as arrows, from one label to another. Take for example the initiate relation in Figure 3.

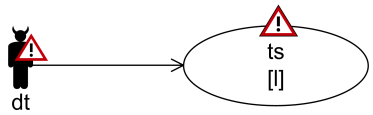

Fig. 3. Initiate relation from a deliberate threat to a threat scenario

Replacing the icon for the deliberate threat with its label gives us dt, and doing the same for the threat scenario gives us ts. The translation of the complete relation is then

$$dt \rightarrow ts.$$

Note that the translation of the threat scenario as a vertex retains the assigned likelihood: $ts(l)$. The translation of a diagram is the pair of the translations of the set of vertices and the set of relations between them.

(B) Translation from the textual syntax into English
In the second 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 vertices, and then move on to the definition of the semantics for the relations. The translation rules of the initiate and leads-to 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 leads-to relations. This can be replaced by either *at* or *nht* for accidental and non-human threats.

Complete threat diagram
A threat diagram $D := (v_1, \ldots, v_n, r_1, \ldots, r_m)$, $n > 0, m \geq 0$, is translated by translating each of its vertices and relations. Note that the logical conjunction implicit in the commas of the sets of vertices and relations translates into a period (we use period instead of comma to increase readability):

$$[\![D]\!] := [\![v_1]\!] \ldots [\![v_n]\!] [\![r_1]\!] \ldots [\![r_m]\!]$$

Vertices

$[\![dt]\!] :=$ ***dt*** is a deliberate threat.
$[\![at]\!] :=$ ***at*** is an accidental threat.
$[\![nht]\!] :=$ ***nht*** is a non-human threat.
$[\![a]\!] :=$ ***a*** is an asset.
$[\![ts]\!] :=$ Threat scenario ***ts*** occurs with undefined likelihood.
$[\![ts(l)]\!] :=$ Threat scenario ***ts*** occurs with $[\![l]\!]$.
$[\![ui]\!] :=$ Unwanted incident ***ui*** occurs with undefined likelihood.
$[\![ui(l)]\!] :=$ Unwanted incident ***ui*** occurs with $[\![l]\!]$.

Initiate relation

$[\![dt \rightarrow ts]\!] :=$ ***dt*** initiates ***ts*** with undefined likelihood.
$[\![dt \xrightarrow{l} ts]\!] :=$ ***dt*** initiates ***ts*** with $[\![l]\!]$.
$[\![dt \xrightarrow{V_n} ts]\!] :=$ ***dt*** exploits $[\![V_n]\!]$ to initiate ***ts*** with undefined likelihood.
$[\![dt \xrightarrow{V_n\ l} ts]\!] :=$ ***dt*** exploits $[\![V_n]\!]$ to initiate ***ts*** with $[\![l]\!]$.

Leads-to relation

$$[\![\, ts_1 \rightarrow ts_2 \,]\!] := \boldsymbol{ts}_1 \text{ leads to } \boldsymbol{ts}_2 \text{ with undefined likelihood.}$$

$$[\![\, ts_1 \xrightarrow{l} ts_2 \,]\!] := \boldsymbol{ts}_1 \text{ leads to } \boldsymbol{ts}_2 \text{ with } [\![\, l \,]\!].$$

$$[\![\, ts_1 \xrightarrow{V_n} ts_2 \,]\!] := \boldsymbol{ts}_1 \text{ leads to } \boldsymbol{ts}_2 \text{ with undefined likelihood, due to } [\![\, V_n \,]\!].$$

$$[\![\, ts_1 \xrightarrow{V_n\ l} ts_2 \,]\!] := \boldsymbol{ts}_1 \text{ leads to } \boldsymbol{ts}_2 \text{ with } [\![\, l \,]\!], \text{ due to } [\![\, V_n \,]\!].$$

Impact relation
As for the two previous relations, the semantics for the unannotated impact relation is the same independent of whether it originates in a threat scenario or an unwanted incident.

$$[\![\, ts \rightarrow a \,]\!] := \boldsymbol{ts} \text{ impacts } \boldsymbol{a}.$$

However, only impact relations originating in unwanted incidents may be annotated with consequences. This relation have the following semantics:

$$[\![\, ui \xrightarrow{c} a \,]\!] := \boldsymbol{ui} \text{ impacts } \boldsymbol{a} \text{ with } [\![\, c \,]\!].$$

Annotations
The following are the translations of the annotations left undefined in the semantics for the relations:

$$\begin{aligned} [\![\, v \,]\!] &:= \text{ vulnerability } \boldsymbol{v} \\ [\![\, V_n \,]\!] &:= \text{ vulnerabilities } \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \\ [\![\, l \,]\!] &:= \text{ likelihood } \boldsymbol{l} \\ [\![\, c \,]\!] &:= \text{ consequence } \boldsymbol{c} \end{aligned}$$

3 Dependent Threat Diagrams

A security risk analysis may target complex systems, including systems of systems. In cases like these we want to be able to

(A) decompose the analysis, such that the sum of the analyses of its parts contributes to the analysis of the composed system, and
(B) compose the results of already conducted analyses of its parts into a risk picture for the system as a whole.

In cases in which there is mutual dependency between the system components, we must be able to break the circularity that the dependency introduces in order to deduce something useful for the composed system. This motivates the introduction of dependent threat diagrams. Dependent threat diagrams are threat diagrams which may also express the property of context dependency. In order to capture context dependencies, we propose some changes to the threat diagram notation introduced above.

3.1 Syntax of Dependent Threat Diagrams

The main difference from the threat diagrams presented above is that dependent threat diagrams distinguish between the context and system scenarios. Figure 4 presents an example of a dependent threat diagram.

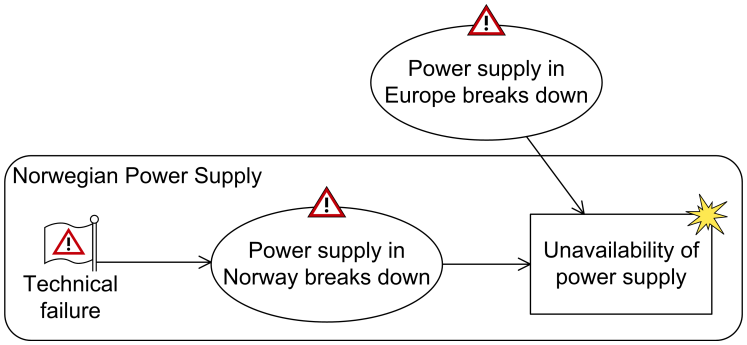

Fig. 4. Dependent threat diagram example

The only modification to the graphical syntax of ordinary threat diagrams is the rectangular container. Everything inside it describes various properties of the system under analysis (i.e. the system scenario), while everything outside captures the assumptions about its context or environment (i.e. the context scenario). In the textual syntax the additional expressiveness is captured as follows:

$$\begin{aligned} \mathit{dependent\ diagram} &= \mathit{context\ scenario} \triangleright \mathit{system\ scenario};\\ \mathit{context\ scenario} &= \mathit{diagram};\\ \mathit{system\ scenario} &= \mathit{diagram}; \end{aligned}$$

Any vertex or relation that is inside the rectangular container belongs to the system scenario; any that is fully outside it belongs to the context scenario. This leaves the relations that cross the rectangular container. For simplicity, we assign these to the system scenario in this paper. However, there may be situations where this is less natural. In the full language we envisage that there will be syntactic means to specify more specialised interpretations of relations crossing the rectangular container.

3.2 Semantics of Dependent Threat Diagrams

We need to define the semantics for the additional syntax of dependent threat diagrams. Let C be a context and S a system scenario, then the semantics for the dependent diagram $C \triangleright S$ is

$$[\![\, C \triangleright S \,]\!] := \text{ If: } [\![\, C \,]\!] \text{ Then: } [\![\, S \,]\!].$$

Context and system scenarios are translated into expressions in English in the same way as ordinary threat diagrams (see Section 2.2).

4 Deduction Rules for Reasoning about Dependency

In order to facilitate the reasoning about dependent threat diagrams we introduce some helpful deduction rules that are meant to be sound with respect to the natural language semantics of dependent threat diagrams.

We say that a relation or vertex R in the system scenario (to the right of the operator $\triangleright$) depends on the context scenario C, denoted $C \ddagger R$, if there is an unbroken path of arrows from a vertex in C to R. For example, in Figure 4, the two relations caused by the threat *Technical failure* are both independent of the context. Formally we define independence as:

Definition 1 (Independence) *Given a diagram $C \triangleright S$:*

1. *A vertex $v \in S$ is independent of the context scenario, written $C \ddagger v$, if any path*
$$(\{v_1, \ldots, v_n, v\}, \{v_1 \rightarrow v_2, v_2 \rightarrow v_3, \ldots, v_n \rightarrow v\}) \subseteq C \cup S$$
is completely contained in $S \setminus C$.
2. *A relation $v_1 \rightarrow v_2$ is independent of the context scenario, written $C \ddagger v_1 \rightarrow v_2$, if its left vertex v_1 is.*

The following rule of independence states that if we have deduced that context scenario C gives relation or vertex R and we have deduced that R is independent from C, then we can deduce R from no context:

Deduction Rule 1
$$\frac{C \ddagger R \quad C \triangleright R}{\triangleright R}\ \ddagger$$

The following rule states that if we have deduced that context scenario C gives system scenario S, and we have deduced that C holds generally, then we can deduce that S holds generally (modus ponens for the operator $\triangleright$):

Deduction Rule 2
$$\frac{C \triangleright S \quad \triangleright C}{\triangleright S}\ \triangleright elim$$

In order to calculate likelihoods propagating through the diagram, we have the following deduction rule:

Deduction Rule 3 *If the vertices $v_1, \ldots, v_n$ represent mutually exclusive events each leading to the vertex v which has undefined likelihood, then*

$$\frac{\triangleright(\{v_1(f_1), \ldots, v_n(f_n), v, V\}, \{v_1 \xrightarrow{l_1} v, \ldots, v_n \xrightarrow{l_n} v, R\})}{\triangleright(\{v(\sum_{i=1}^{n} f_i \cdot l_i)\}, \{\})}\ mutex$$

where V, R are without occurrences of the vertex v.

5 Example Case – Mutual Dependent Sticks

We will address a simple case of mutual dependency that represents the core issue targeted by dependent CORAS diagrams, namely the scenario that two sticks lean against each other such that if one stick falls over, the other stick will fall too. Thus, the likelihood that Stick2 will fall given that Stick1 has fallen is 1, as for the reverse order. We want to operate with different likelihood estimates as to how often each of them will fall so that we can see how the various measures in the component analysis affect the likelihood estimates of the overall system. Therefore, we imagine that Stick1 is likely to fall due to strong wind, while Stick2 is placed in front of a shield which minimizes the likelihood of it being subject to the same threat scenario. However, Stick2 is attractive to birds who like to seek shelter from the wind, many at a time. When the number of birds sitting on it is high, the stick may fall due to the pressure. Birds do not prefer the windy spot, and consequently Stick1 is not subject to this risk. As illustrated by the dependent threat diagrams in Figure 5 and 6, we estimate the likelihoods of the two scenarios to be two times per year and once a day respectively.

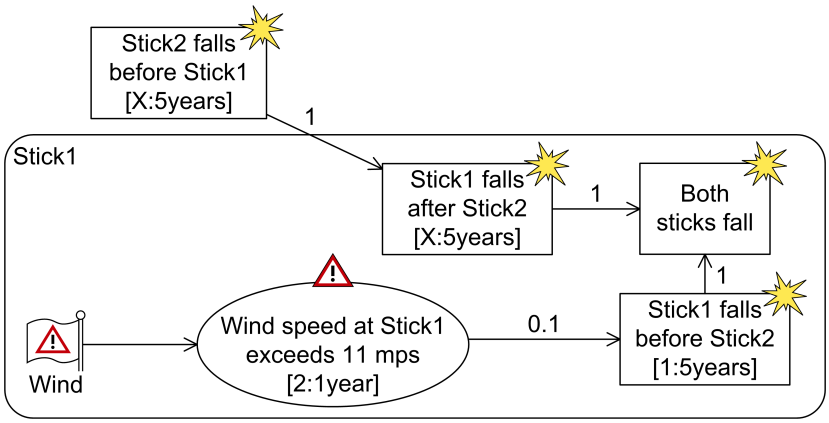

Fig. 5. Threat diagram for Stick1

There are two possibilities for both sticks falling.[4] Either Stick2 falls first and then Stick1 falls or the other way around. This means that the likelihood that Stick1 falls depends on the likelihood of whether Stick2 falls and vice versa.

[4] For simplicity we ignore the case that both sticks fall at the same time.

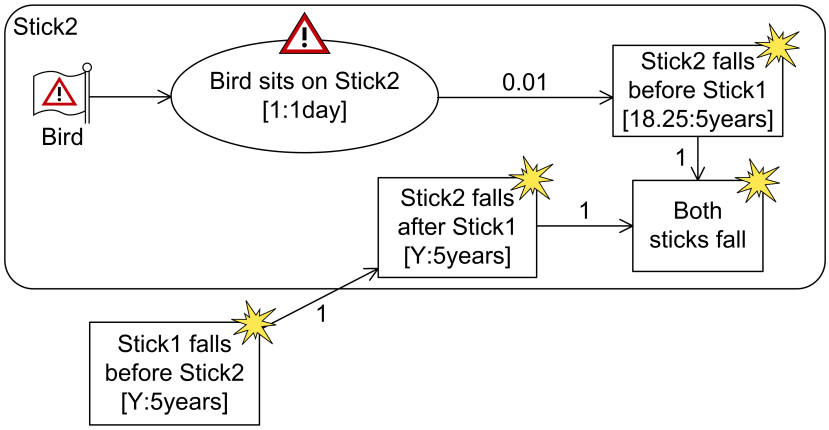

Fig. 6. Threat diagram for Stick2

5.1 Using the Deduction Rules on the Case

Using the following abbreviations

B = Bird	Bon2 = Bird sits on Stick2
W = Wind	Wat1 = Wind speed at Stick1 exceeds 11 mps
2b1 = Stick2 falls before Stick1	1a2 = Stick1 falls after Stick2
1b2 = Stick1 falls before Stick2	2a1 = Stick2 falls after Stick1
1&2 = Both sticks fall	

we get the following representations of the diagrams in the textual syntax:

$$
\begin{aligned}
\text{Stick1} = {} & (\{2\text{b1}(x{:}5yr)\}, \{\}) \triangleright \\
& (\{\text{W}, \text{Wat1}(2{:}1yr), 1\text{b2}(1{:}5yr), 1\text{a2}(x{:}5yr), 1\&2\}, \\
& \{2\text{b1} \xrightarrow{1} 1\text{a2}, \text{W} \rightarrow \text{Wat1}, \text{Wat1} \xrightarrow{0.1} 1\text{b2}, 1\text{b2} \xrightarrow{1} 1\&2, 1\text{a2} \xrightarrow{1} 1\&2\}) \\
\text{Stick2} = {} & (\{1\text{b2}(y{:}5yr)\}, \{\}) \triangleright \\
& (\{\text{B}, \text{Bon2}(365{:}1yr), 2\text{b1}(18.25{:}5yr), 2\text{a1}(y{:}5yr), 1\&2\}, \\
& \{1\text{b2} \xrightarrow{1} 2\text{a1}, \text{B} \rightarrow \text{Bon2}, \text{Bon2} \xrightarrow{0.01} 2\text{b1}, 2\text{b1} \xrightarrow{1} 1\&2, 2\text{a1} \xrightarrow{1} 1\&2\})
\end{aligned}
$$

5.2 Composing Dependent Diagrams

Using the deduction rules and the dependent threat diagrams for the two components, we deduce the validity of the threat diagram for the combined system

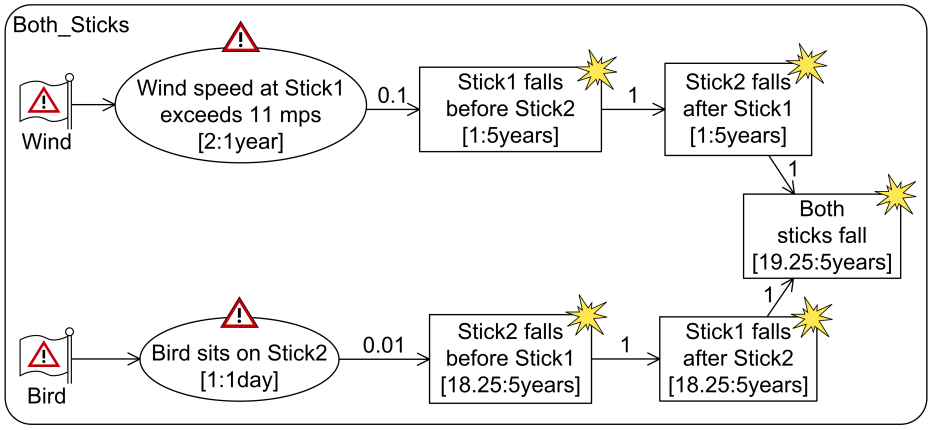

Fig. 7. Threat diagram for the combined system

Both_Sticks, presented in Figure 7:

$$\begin{aligned}\text{Both\_Sticks} \;=\; \triangleright\;&(\{\mathrm{W}, \mathrm{Wat1}(2{:}1yr), \mathrm{1b2}(1{:}5yr), \mathrm{2a1}(1{:}5yr), \mathrm{B}, \mathrm{Bon2}(365{:}1yr),\\ &\mathrm{2b1}(18.25{:}5yr), \mathrm{1a2}(18.25{:}5yr), \mathrm{1\&2}(19, 25{:}5yr)\},\\ &\{\mathrm{W} \rightarrow \mathrm{Wat1}, \mathrm{Wat1} \xrightarrow{0.1} \mathrm{1b2}, \mathrm{1b2} \xrightarrow{1} \mathrm{2a1}, \mathrm{2a1} \xrightarrow{1} \mathrm{1\&2}, \mathrm{B} \rightarrow \mathrm{Bon2},\\ &\mathrm{Bon2} \xrightarrow{0.01} \mathrm{2b1}, \mathrm{2b1} \xrightarrow{1} \mathrm{1a2}, \mathrm{1a2} \xrightarrow{1} \mathrm{1\&2}\})\end{aligned}$$

The proof that the diagrams in Figure 5 and Figure 6 give the diagram in Figure 7 goes as follows[5]. For readability and to save space we have only included the probability annotations for the threat scenarios and unwanted incidents 2b1, 1a2, 1b2, 2a1 and 1&2 in the proof, because these are the only ones that have variables as parameters or whose values are undefined in one of the diagrams. Their likelihood values are therefore affected by the proof.

$\langle 1\rangle 1$. ASSUME: 1. Stick1
2. Stick2
PROVE: Both_Sticks

$\langle 1\rangle 2$. $\triangleright(\{\mathrm{W}, \mathrm{Wat1}, \mathrm{1b2}(1{:}5yr)\}, \{\mathrm{W} \rightarrow \mathrm{Wat1}, \mathrm{Wat1} \xrightarrow{0.1} \mathrm{1b2}, \mathrm{1b2} \xrightarrow{1} \mathrm{1\&2}\})$
PROOF: By assumption $\langle 1\rangle 1$, Deduction Rule 1 and predicate logic.

$\langle 1\rangle 3$. $\triangleright(\{\mathrm{B}, \mathrm{Bon2}, \mathrm{2b1}(18.25{:}5yr)\}, \{\mathrm{B} \rightarrow \mathrm{Bon2}, \mathrm{Bon2} \xrightarrow{0.01} \mathrm{2b1}, \mathrm{2b1} \xrightarrow{1} \mathrm{1\&2}\})$
PROOF: By assumption $\langle 1\rangle 1$, Deduction Rule 1 and predicate logic.

$\langle 1\rangle 4$. $(\{\mathrm{1b2}(1{:}5yr)\}, \{\}) \triangleright (\{\mathrm{2a1}(1{:}5yr), \mathrm{1\&2}\}, \{\mathrm{1b2} \xrightarrow{1} \mathrm{2a1}, \mathrm{2a1} \xrightarrow{1} \mathrm{1\&2}\})$
PROOF: By instantiation of the free variables in assumption $\langle 1\rangle 1$. Stick2.

$\langle 1\rangle 5$. $(\{\mathrm{2b1}(18.25{:}5yr)\}, \{\}) \triangleright (\{\mathrm{1a2}(18.25{:}5yr), \mathrm{1\&2}\}, \{\mathrm{2b1} \xrightarrow{1} \mathrm{1a2}, \mathrm{1a2} \xrightarrow{1} \mathrm{1\&2}\})$

[5] The proof is written in Lamport's Latex style for writing proofs [14].

PROOF: By instantiation of the free variables in assumption $\langle 1 \rangle 1$. Stick1.

$\langle 1 \rangle 6$. $\triangleright(\{2a1(1:5yr), 1\&2\}, \{1b2 \xrightarrow{1} 2a1, 2a1 \xrightarrow{1} 1\&2\})$

PROOF: By $\langle 1 \rangle 2$, $\langle 1 \rangle 4$, Deduction Rule 2 and predicate logic.

$\langle 1 \rangle 7$. $\triangleright(\{1a2(18.25:5yr), 1\&2\}, \{2b1 \xrightarrow{1} 1a2, 1a2 \xrightarrow{1} 1\&2\})$

PROOF: By $\langle 1 \rangle 3$, $\langle 1 \rangle 5$, Deduction Rule 2 and predicate logic.

$\langle 1 \rangle 8$. $\triangleright(\{2a1(1:5yr), 1a2(18.25:5yr), 1\&2(19.25:5yr)\},$
$\{1b2 \xrightarrow{1} t, 2a1 \xrightarrow{1} 1\&2, 2b1 \xrightarrow{1} 1a2, 1a2 \xrightarrow{1} 1\&2\})$

PROOF: By $\langle 1 \rangle 6$, $\langle 1 \rangle 7$, Deduction Rule 3, since 2a1 and 1a2 represent mutually exclusive events, and predicate logic.

$\langle 1 \rangle 9$. Q.E.D.

PROOF: By $\langle 1 \rangle 2$, $\langle 1 \rangle 3$ and $\langle 1 \rangle 8$.

6 Conclusion

We have argued that dependent threat diagrams may serve to analyse the circularity of the classical mutual dependency case. The approach may be applied to situations in which we may decompose the target of analysis, and perform a risk analysis on each component. Given that all parts of the system are analysed we may construct a composed analysis in which dependency estimates appear due to the results in each component analysis. The property of decomposition is useful also because it enables the reuse of results from the analysis of a system component in the analysis of any system containing it.

6.1 Related work

The CORAS language originates from a UML [18] profile [16, 19] developed as a part of the EU funded research project CORAS[6] (IST-2000-25031) [1]. The CORAS language has later been customised and refined in several aspects, based on experiences from industrial case studies, and by empirical investigations documented in [5, 6, 7]. Misuse cases [3, 21, 22] was an important source of inspiration in the development of the UML profile mentioned above. A misuse case is a kind of UML use case [11] 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 [15]. 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) [9]. 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) [8] 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

[6] `http://coras.sourceforge.net`

trees [20] 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). The separation of diagrams into a context scenario and a system scenario is inspired by the assumption-guarantee paradigm used to facilitate modular reasoning about distributed systems. See [12, 17] for the original proposals, and [2] for a more recent treatment. We are not aware of other approaches to risk analysis or threat modelling that are based on the assumption-guarantee paradigm.

6.2 Acknowledgements

The research for this paper has been partly funded by the DIGIT (180052/S10) and COMA (160317) projects of the Research Council of Norway.

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] Martin Abadi and Leslie Lamport. Conjoining specifications. *ACM Transactions on programming languages and systems*, 17(3):507–534, 1995.

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

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

[6] 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.

[7] 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.

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

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

[10] ISO/IEC 14977:1996(E). *Information Technology — Syntactic Metalanguage — Extended BNF*, first edition, 1996.

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

[12] Cliff B. Jones. *Development Methods for Computer Programmes Including a Notion of Interference.* PhD thesis, Oxford University, UK, 1981.

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

[14] Leslie Lamport. How to write a proof. Technical report, Digital Systems Research Center, 1993.

[15] 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.

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

[17] Jayadev Misra and K. Mani Chandy. Proofs of networks of processes. *IEEE Transactions on Software Engineering*, 7(4):417–426, 1981.

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

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

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

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

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