D5.2 Documentation of forecasts of future evolvement

Mass Soldal Lund, Bjørnar Solhaug, Ketil Stølen (SIN), Ruth Breu, Michael Breu, Frank Innerhofer-Oberperfler (UIB), Edith Felix, Benjamin Fontan (THA), Alessandra Tedeschi (DBL), Elisa Chiarani (UNITN)

Document information

Document Number	D5.2
Document Title	Documentation of forecasts of future evolvement
Version	1.0
Status	Final
Work Package	WP 5
Deliverable Type	Report
Contractual Date of Delivery	31 January 2010
Actual Date of Delivery	25 January 2010
Responsible Unit	SIN
Contributors	SIN, THA, UIB, DBL, UNITN
Keyword List	Evaluation, language, risk analysis, evolving systems
Dissemination level	PU

Document change record

Version	Date	Status	Author (Unit)	Description
0.1	14.09.09	Draft	Mass Soldal Lund (SIN)	Outline
0.2	30.10.09	Draft	Mass Soldal Lund (SIN), Alessandra Tedeschi (DBL)	Updated outline, first draft validation scenario
0.3	07.11.2009	Draft	Mass Soldal Lund, Ketil Stølen (SIN), Ruth Breu, Michael Breu (UIB), Edith Felix, Benjamin Fontan (THA), Alessandra Tedeschi (DBL)	Updated structure First draft of meta-model Updated validation scenario
0.4	16.11.2009	Draft	Michel Breu (UIB)	Draft of Section 4.3
0.5	04.12.2009	Draft	Benjamin Fontan, Edith Felix (THA)	Draft of Section 4.2 Extensions to Security DSML, Update of Section 3.1 Alignment of Approach
0.6	22.12.1009	Draft	Mass Soldal Lund, Bjørnar Solhaug, Ketil Stølen (SIN)	Restructuring and new contents in Sections 2 and 3
0.7	24.01.2010	Pre-final	Mass Soldal Lund, Bjørnar Solhaug, Ketil Stølen (SIN), Elisa Chiarani (UNITN), Frank Innerhofer-Oberperfler (UIB)	First quality check, Internal review, Removed Section 5 (validation), Added Executive summary and Sections 1, 4.1 and 5.
1.0	25.01.2010	Final	Mass Soldal Lund, Ketil Stølen (SIN), Elisa Chiarani (UNITN)	Final quality check completed, minor corrections, proof reading

Executive Summary

A risk analysis typically focuses on a particular configuration of the target at a particular point in time, and is valid under the assumptions made in the analysis. However, both the risk analysis target and its environment can change and evolve over time. We therefore need methods and techniques to reflect such changes in the risk analysis. This deliverable is concerned with the development of modelling support for risk analysis of changing and evolving systems; in other words, language support for modelling a changing and evolving risk picture.

How we handle changes in a risk analysis depends to a large degree on the context and the types of changes we are dealing with: Are the changes the results of maintenance or of bigger, planned changes? Are the changes a transition from one stable state of the target to another or the continuous evolution of a target designed to change over time? Do the changes occur in the target or in the environment of the target? The answers to such questions decide how we handle the changes.

In a conceptual clarification of the domain of risk analysis of changing and evolving systems three perspectives of change have been identified:

- *The maintenance perspective*, where an old risk picture is updated after the old analysis target has been updated.
- *The before-after perspective*, where a risk picture of future changes of an analysis target is made based on a risk picture of the current analysis target.
- *The continuous evolution* perspective, where a risk picture is generalized to capture evolution of risks as the analysis target evolves.

For each of these perspectives a work process, a conceptual model and language requirements are developed. Based on this, generic language extensions that provide language support for each of the perspectives are formalized. The conceptual model, the language requirements and the generic language extensions are developed in a general manner and such that they may be applied to a large range of existing languages for risk modelling.

The proposed approach is then demonstrated by instantiating the general language support for three concrete risk modelling languages. This way we obtain three concrete risk modelling languages extended with full or partial support for risk analyses of changing and evolving systems. One of the languages is extended by instantiated the generic language extensions, while two of the languages are extended by refining and instantiating the conceptual models.

Index

1 Introduction

A risk analysis typically focuses on a particular configuration of the target at a particular point in time, and is valid under the assumptions made in the analysis. However, both the risk analysis target and its environment can change and evolve over time. We therefore need methods and techniques to reflect such changes in the risk analysis. This deliverable is concerned with the development of modelling support for risk analysis of changing and evolving systems. In other words: language support for modelling a changing and evolving risk picture. This language support is given in the form of generic language extensions that may be instantiated in existing approaches to risk modelling.

How the system that is the target of analysis changes and evolves will obviously influence the changing and evolving risk picture that we seek to capture. However, system analysis and system modelling is the concern of Work Package 4 of the SecureChange project and this deliverable does not provide the language support for modelling the changing and evolving system.

We relate to the system analysis and modelling work of Work Package 4 – and indirectly to the work on changing and evolving requirements of Work Package 3 – by making high level assumptions concerning the system modelling and by defining mappings that connect our models of the risk picture to system models. The relations between the work of Work Packages 3, 4 and 5 are illustrated in Figure 1.

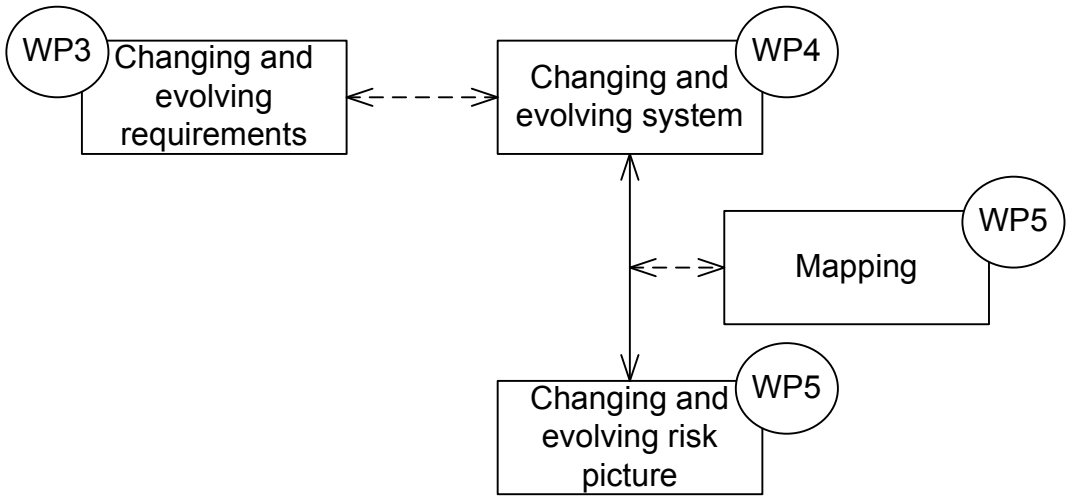

Figure 1 Relations between WP3, WP4 and WP5

In this deliverable we attack the challenges of modelling a changing and evolving risk picture from a predominantly top-down viewpoint. We start out by providing conceptual clarifications in Section 2, where we characterize and clarify the domain what we are working in. In that section we discuss different perspectives on change within the domain of risk analysis and different kinds of change that can affect the analysis results.

In Section 3 we go into detail on the perspectives. For each of them we sketch the work process of conducing risk analysis within the perspective and provide a conceptual model for the perspective. Based on the work process and the conceptual model we specify requirements to the language support of the perspective and formalize generic language extensions by means of an abstract syntax.

The idea is that, given a risk modelling language, the language support for a perspective can be obtained by implementing the generic language extension in that particular language. In Section 4 we demonstrate this with three different risk modelling languages: the CORAS language, Security DSML and ProSecO. With respect to the CORAS language, the implementation of the generic language extensions is done by means of instantiating the abstract syntax of the language extension. With respect to Security DSML and ProSecO, the implementation of the language extensions is obtained by refining and instantiating the conceptual models of the perspectives.

In Section 5 we provide conclusions and in the appendix we provide a glossary of risk analysis terms.

2 Conceptual Clarification

How we handle changes in a risk analysis depends to a large degree on the context and the types of changes we are dealing with: Are the changes the results of maintenance or of bigger, planned changes? Are the changes a transition from one stable state of the target to another or the continuous evolution of a target designed to change over time? Do the changes occur in the target or in the environment of the target? The answers to such questions decide how we handle the changes. We therefore start by presenting three perspectives of change in risk analyses in Section 2.1, and different kinds of change relevant for risk analyses in Section 2.2. In Section 2.3 we summarize the state-of-the-art on risk modelling presented in [18].

2.1 Perspectives on Change

As stated above, the context of the changes is of importance for what kind of approach we choose for dealing with the changes in risk analysis. There are two dimensions to what we define as the change perspective. The first is whether the change was planned or not, i.e. if the risk analysis is pro- or re-active. The second dimension is captured by the concepts of evolution and revolution:

- *Evolution:* Smaller changes that accumulate over time. Bug fixes and upgrades of computer systems are typically an evolution.
- *Revolution:* Major changes that have large effects on the target. The rollout of a new system is a typical example of a revolution.

Using these two dimensions, we identify three different viewpoints or perspectives on change:

1. *The maintenance perspective (a posteriori perspective)*: Sometimes the target evolves over time, changes accumulate unnoticed, and risk analysis documentation and results may become outdated. An outdated risk analysis may give a false picture of the risks associated with the target and when changes occur we may need to conduct a new risk analysis. Conducting a risk analysis from scratch is expensive and time-consuming, and we would rather like to update the documentation from the risk analysis that we have already conducted. In terms of the dimensions defined above, the maintenance perspective is a reactive evolution.
2. *The before-after perspective (a priori perspective)*: We often plan and anticipate changes, and major changes to the target may even be the motivation for a risk analysis. Such planned changes require special treatment for two reasons: First, it is very important to have a clear understanding of what characterizes the target “as-is” and what characterizes the target “to-be”, and of what are the differences between these two. Second, the process of change may itself be a source of risks. In terms of the perspective dimensions, before-after is proactive revolution.

3. *The continuous evolution perspective*: There may be cases where we plan for the target to evolve over time or where we can anticipate gradual changes, e.g. if we plan to gradually increase the number of components working in parallel, if we plan to gradually include more and more sites into a system if we anticipate a gradual wear of hardware over time, or if we foresee an increase in users of a system or the number of attacks by an adversary. What is common to such cases is that the target can be described as a function of time. Obviously then, it would be a benefit if we could also do a risk analysis that is a function of time. Such a risk analysis would give a risk picture not for one or a few, but for any future point in time. In terms of the perspective dimensions, the continuous evolution perspective is proactive evolution.

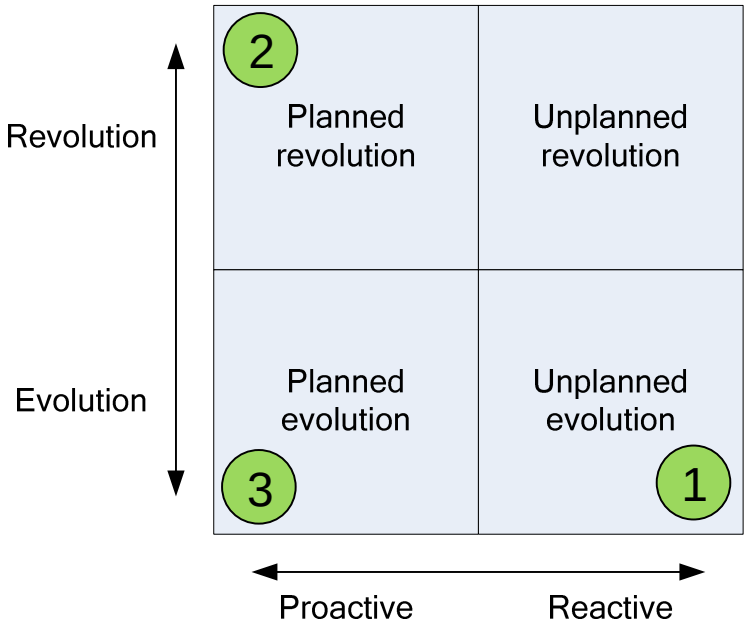

Figure 2 Relation between dimensions and perspectives

The relation between the dimensions and perspectives are illustrated in Figure 2. When it comes to the last combination of the perspective dimensions, reactive revolution, this would be a large unforeseen change that necessitates a completely new risk analysis, and is therefore not considered further in this document.

2.2 Kinds of Change

In a risk analysis, information is collected and organized to describe the target of analysis and its environment. The scope and focus of the analysis is furthermore characterized, defining the parts of the system that are most relevant to the analysis. Together, such information serve as the input to the analysis, and the risk analysis results are derived from this input through a risk analysis process. Any change in this information may therefore cause changes in the outcome of the risk analysis. We refer collectively to the information that forms the input to a risk analysis as the *target description*. The class diagram of Figure 3 shows the elements of the target description

and the relations between them. In this subsection we explain each of these elements, and we also explain how changes to any of them may result in changes in the resulting risk picture.

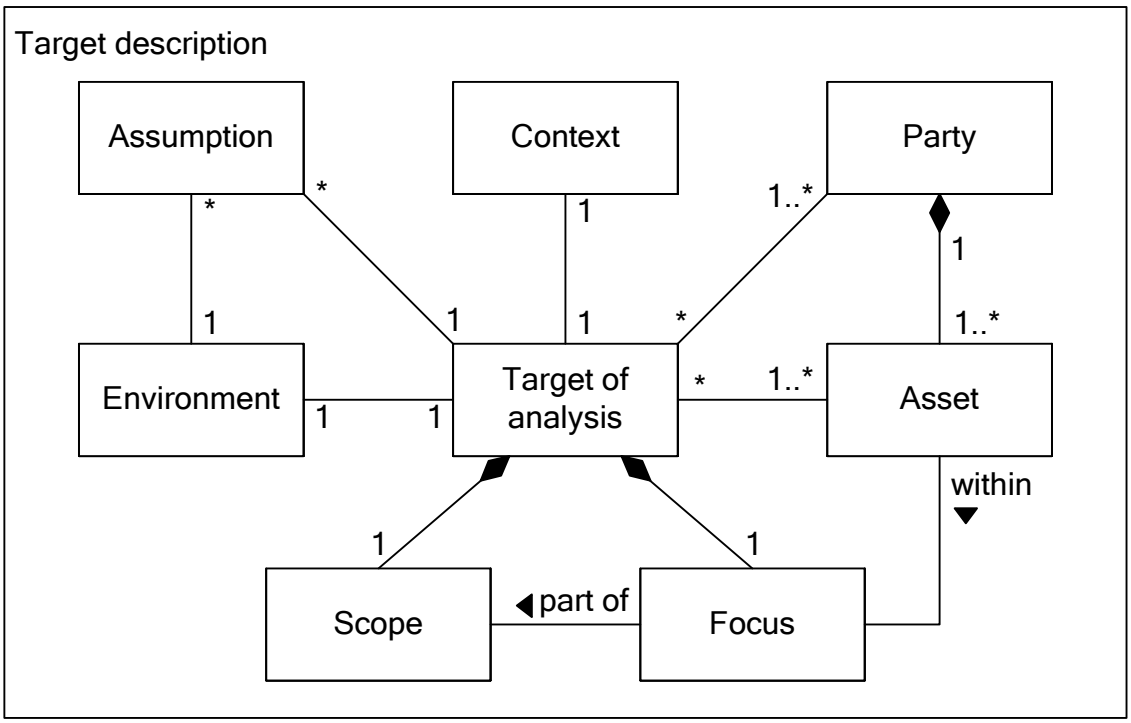

Figure 3 Target description

In addition to changes in the target description a current risk picture may need to be updated because we have gained new or better information about the target or its environment. Taking this into account we can therefore distinguish between two main categories of change, namely (i) changes in the target description and (ii) changes in our knowledge about the target and its environment. In the following, we take a closer look at relevant kinds of changes of the target description in Section 2.2.1 and at changes in our knowledge in Section 2.2.2. Moreover, the process of changing the target may itself be a source of risks, which is discussed in Section 2.2.3.

2.2.1 Changes in the Target Description

2.2.1.1Target of Analysis

The target of analysis is the system, organization, enterprise, etc. or parts thereof, that is the subject of the analysis. Changes to the target must be expected, even in what we would consider a stable system. Consider for example bug fixes distributed from third party software vendors. Another obvious example of changes to the target is implementation of treatments identified in a risk analysis. But changes may also be more extensive, such as introduction of new functionality in a software system or replacement of software components, work processes or hardware components. We allow full generality when defining the target, and changes to the target may be as general as the target itself. It is therefore necessary to characterize in more detail what

changes to the target may constitute. In all, we distinguish between four main kinds of changes to the target of analysis:

1. *Changes to the functions/functionality of the target:* This represent changes to all physical or logical parts of the target that exhibit relevant behaviour. This may be computer hardware and software, but also mechanical and moving parts.
2. *Changes to the non-functional properties of the target:* This includes, among other things, changes to security mechanisms and safety systems, and introduction of barriers.
3. *Changes to the processes of the target:* There are often work processes associated with the target. These may be of equal importance to the risk analysis as the components of the target, and changes to the processes must be considered changes to the target. Such changes also include organisational changes that may be of relevance.
4. *Changes in policies associated with the target:* Policies restrict the functionality and the processes of a system. This means that changes in policies may be of equal relevance to the risk analysis as changes to the components or the processes of the target.

2.2.1.2 Scope and/or Focus

The scope of the analysis defines the extent or range of the target of the analysis. The scope defines the border of the analysis, i.e. what is held inside of and what is held outside of the analysis, or in other words, what is considered to be part of the target of analysis and what is considered to be part of the target's environment. The focus, on the other hand, defines the main issue or central area of attention in the risk analysis. It is clear that the focus is defined within the scope of the analysis.

Any change in the focus and/or scope will almost inevitably result in changes in the risk analysis and thereby outdate any existing risk analysis results and documentation. Change of scope or focus may obviously be relevant also for cases in which the system or environment in question has not undergone changes, since a client or other stakeholders may wish to gain knowledge of the risks in a wider setting or with a different focus. However, change of focus or scope is particularly relevant for changing and evolving systems. If functionality is changed, services are substituted, new groups of end-users emerge, other interfaces are introduced, etc., it may often be that the focus and/or the scope of the analysis must be reconsidered in order to derive a more relevant and up to date risk picture.

2.2.1.3 Environment

It is not only changes to the target of analysis itself that may affect and outdate risk analysis documentation and results. There can be changes to the world outside the boundaries of the target that might be of equal or even greater relevance for the risk picture of the target.

The environment of the target may be anything in the surroundings of the target that is relevance and that may affect or interact with the target; in the most general case the environment is the rest of the world. One specific change of the environment is that a

new kind of threat emerges or that a threat disappears or is no longer relevant for the risk analysis. Obvious examples of new threats (in a computer security setting) are the invention of new kinds of computer viruses or hacker attacks. On a higher level, the emergence of electronic warfare and cyber crime are other examples.

Another kind of change in the environment is changes in the likelihood of threat scenarios due to changes in external factors. An example of this is threat scenarios involving blackouts. The likelihood of such threat scenarios may be dependent on stability of external power supply, so if there are changes in the reliability of the external power supply, the likelihood of the threat scenarios might change.

2.2.1.4 Assumptions

Sometimes it is not changes to the target or its environment that triggers the need for changes in the risk analysis results, but changes to the assumptions made in the analysis. There are several reasons why we might want to change the assumptions after completion of a risk analysis, and most often changes in the assumptions means that we also do changes to the scope of the analysis. It might be that parts of a system were assumed to be secure and for that reason kept outside the target of the analysis, but that we later get evidence for the contrary (or for other reason start to doubt the validity of the assumption) and therefore want to include them in the target. Changes in assumptions are particularly relevant for changing and evolving systems since certain assumptions may have to be discarded after the introduction or replacement of functionality, services, end-users, etc.

2.2.1.5 Parties and/or Assets

A party of a risk analysis is an organization, company, person, group or other party on whose behalf the risk analysis is conducted. Each risk that it identified and assessed is associated with a party, so any change of parties will directly affect the risk analysis results. More specifically, each risk is associated with an asset, and an asset is something to which a party assigns value and hence for which the party requires protection.

There are two additional ways in which changes in parties may be relevant in a risk analysis. First, there may be organizational changes with respect to the customer of an analysis as a result of changes in parties. An example might be that the company for which a risk analysis was conducted is bought by another company, and the new owners have different priorities. Second, we may want to use an earlier conducted risk analysis as a template or pattern for later risk analyses. This may be the case if we are doing a risk analysis of a system or organization similar to (or even the same as) earlier analysed targets, or if we are doing a risk analysis in a very similar domain. In this case we may think of it as a risk analysis parameterized with party that we apply as a template or a pattern.

Since an asset is directly related to a party, any changes of parties will result in changes of assets. Irrespective of changes in parties, however, it may also be that the value of an asset is reassessed, an asset is completely removed from the target (for example because it is transferred to another party, or because the new asset value equals zero), or new assets are introduced.

2.2.1.6Context

The context of the analysis is the premises for and background of the analysis; this includes the purposes of the analysis and to whom the analysis is addressed. The context of the analysis is the premises for and the background of the analysis, including the purposes of the analysis and to whom the analysis is addressed. Since changes in the context moreover may require changes in the target description, also the context must be taken into account when dealing with change in risk analyses.

2.2.2 Changes in our Knowledge

The final type of change that can affect our risk analysis results and that we therefore must consider is the possibility of changes in our knowledge about the target and its environment. Risk analysis results are usually dependent on expert opinions and estimated likelihood and consequence values. If we get new or better knowledge about the target or its environment, for example through monitoring, we might want to change our estimates to correspond to this updated knowledge. Changes in our knowledge may also reveal for us new threats and threat scenarios.

When conducting a risk analysis we choose the focus and scope, we make the assumptions, and we make decisions about the parties and the context of the analysis. These things are therefore not subject to observations and empirical investigation in the same sense as the target its environment. Changes in our knowledge about the target and its environment may nevertheless substantially affect the decisions and assumptions upon which a risk analysis is based. The complete target description must therefore be considered for changes and updates in case of changes in our knowledge.

2.2.3 Process of Change

When dealing with larger, planned changes there is another important aspect of the change we need to handle – the process of change itself. In the transition from its old to its new state, the target may be particularly vulnerable to threats, and risks may originate from the changes of the target themselves. In these cases we should also consider doing a risk analysis of the change process itself in addition to a risk analysis of the new state of the target. This is particularly relevant for the before-after perspective on change.

2.3 Summary of State-of-the-Art

In the deliverable D5.1 [18], a number of risk modelling languages were evaluated with respect to the challenges of supporting risk analyses in the three perspectives described in Section 2.1. The purpose of the evolution was to identify the starting point, as well as useful approaches and ideas, for the development of the language.

Several (but far from all) of the languages investigated have some support for associating elements of risk models to parts of the target description. Such a mapping between risk models and target descriptions is relevant in the analysis of changing systems since it enables risk documentation to be traced to the relevant system

documentation, and thereby facilitates the updating of the risk picture following system changes.

- UML based approaches such as mis-use cases [19][20][21] may utilize built in mechanisms in the UML for relating elements from different UML diagram, or a suitable profile for doing so may be defined.
- In the Security DSML [14], architectural components and (security) information are modelled and both security needs and risks are associated with these components. Risk reduction components are related to risks and security objectives. This means that risks are related to parts of the target, but it is at the cost of not having relations between risks and other elements of risk analysis.
- In ProSecO [11], risks are related to elements of a functional model of the target. In addition the model elements are related to security objective and security requirements and risks are related to threats and security controls (for treatment and mitigation).
- Tropos [2][3][8] is mainly a language for modelling and decomposition of goals. Events (unwanted incidents) may be related to the goals, and the events may be decomposed similar to the top event of a fault three. In addition, treatments, represented as tasks, can be associated with the events of the event threes.
- CORAS [4][5][6] has support for modularizing risk models, which means that each module of the model may be associated with a part of the target description.
- In ADONIS [1], risks can be associated with the activities of a business model.

The only approach with some support for modelling states or phases of a change is ProSecO. All elements of a ProSecO security model (security objectives, security requirements, functional elements and risks) have a state which gives the status of the element. When the models change, elements of the model may be transferred to states that indicate that additional risk analysis is needed. ProSecO also gives some support to modelling of the change process by means of state machines that define the state changes of the elements of the security model.

Several languages exist for modelling of processes in general (e.g. UML, SPEM, BPML) and we can assume that they to some degree may be applied for modelling of change processes. However, the only approach we are aware of that allows you to assign risk related information to the processes is ADONIS.

None of the languages for risk modelling investigated have any support for defining an evolving risk picture or for incorporating time into the risk models, since structurally they are all every static. Some of them, however, provide support for updating qualitative values annotated to the diagrams, in the sense that by changing the input values, the derived output values can be automatically updated. These languages include fault trees, Markov models, Bayesian networks and the CORAS language.

To briefly summarise the state of the art, there do exist risk modelling languages that provide some support for doing risk analysis of changes in the maintenance perspective and the before-after perspective. This is not a surprise, as doing two risk analyses and making two risk pictures, one “before picture” and one “after picture” is always an options. For the continuous perspective, however, very little support was found.

3 Generic Language Extensions

In this section we characterize the language support needed in the context of risk analysis of changing targets. As established in our state-of-the-art survey, several risk modelling languages are in existence (see Section 2.3 and also [18]). We do therefore not set out do define a completely new risk modelling language, but rather to define the language extensions required for existing languages to be applied in the context of risk analysis with change. Also for this reason, the language extensions are characterized in a general manner and with the presumption that the proposed extensions will be applicable in a number of different risk modelling languages. In Section 4, we demonstrate this by implementing the general language extensions in different concrete languages.

In line with this general approach, we will in the following assume risk analysis, system analysis and requirement analysis, including their language support, to be black-boxes in the sense that they are well-understood domains and that established concepts and terminology can be inherited from these domains. The black-boxes that we operate with in this report and their relations are shown in the UML class diagram of Figure 4. We presume the existence of risk models that are obtained from risk analyses and the existence of system models that are obtained from system analyses. Further, we presume the existence of requirement models obtained by requirement analyses and consider such models as part of system models. It should be noted that we in this deliverable assume that a system model can capture all elements of a target description as defined in Figure 3 in Section 2. This means we assume the system models can capture assets, assumptions, environment, etc.

In the SecureChange project, Work Package 5 deals with risk analysis and risk models, and this deliverable specifically deals with risk models. System analysis and modelling are topics of Work Package 4 and requirement analysis and modelling are topics of Work Package 3. These topics are therefore outside the scope of this deliverable, but are present as black-boxes in order to emphasize the relations of Work Package 5 to Work Packages 3 and 4.

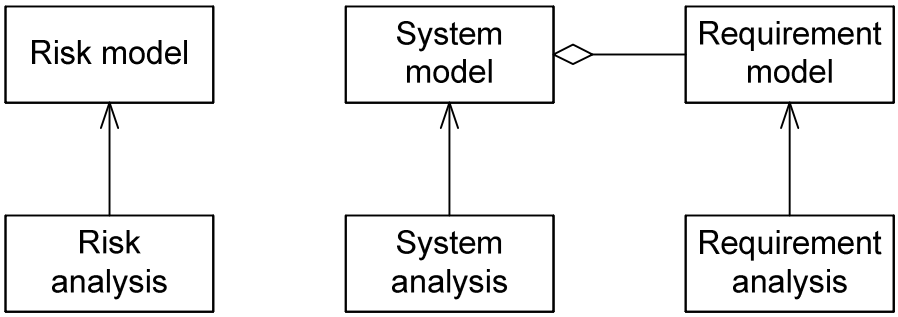

Figure 4 Black-boxes and their relations

An implication of this general approach is that the implementation of the proposed language extensions in a concrete language will involve substituting concrete approaches for these black-boxes. For example, if fault tree analysis (FTA) [10] is the

approach to which we wish to apply our language extensions, we will substitute “Fault tree analysis” and “Fault tree” for the general terms “Risk analysis” and “Risk model”.

In Section 2, three perspectives of change in risk analysis were presented. These three perspectives are shown in a UML class diagram in Figure 5. In the following we have a closer look at each of the perspectives, and decide for each of them the language support needed for doing risk analysis within the perspective.

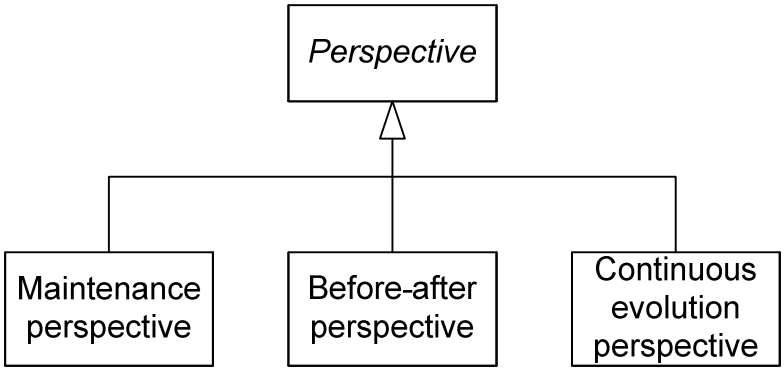

Figure 5 Perspectives on change

When discussing the language support for each of the perspectives we start by sketching a work process by which a risk analysis within the perspective will proceed.[1] The class diagram of Figure 5 serves as the staring point for a conceptual model of risk analysis with change. After presenting the work process of a perspective we continue to elaborate a conceptual model of the perspective. Then, based on the work process and the conceptual model, we define the generic language extensions that support risk analysis within the perspective. The language extensions are defined by means of requirements, but also by an abstract syntax characterizing the language constructs needed in a minimal language that fulfils the requirements. The idea is that a specific approach to modelling risks with change and evolution should instantiate and possibly extend the abstract syntax given for each of the perspectives.

The abstract syntax is introduced in Section 3.1 by characterizing the assumptions we make with respect to risk models and system models in general. The maintenance perspective is discussed in Section 3.2, the before-after perspective in Section 3.3 and the continuous evolution perspective in Section 3.4.

3.1 Abstract Syntax

We define an abstract syntax for risk models and system models. The idea is that this abstract syntax characterizes the assumptions we make about the models and that a specific approach to risk or system modelling should be an instantiation of this abstract syntax.

[1] Methodology for risk analysis within the context of change and evolution is the topic for the next deliverable of Work Package 5 in the SecureChange project “D5.3 Assessment method”. The work processes will be refined and elaborated in this deliverable.

We define a risk model as a set of risk model elements and a set of risk model relations:

$$risk\ mod := (\{risk\ mod\ elem\}, \{risk\ mod\ rel\})$$

A risk model element consists of a risk model element type and a set of attributes:

$$risk\ mod\ elem := (risk\ model\ elem\ type, \{attribute\})$$

Typically, the risk model element types are entities like **risk**, **threat**, and **vulnerability**. The attributes are usually entities like descriptions, identifiers, and likelihood and risk values.

A risk model relation is a mechanism for relating risk model elements. We assume that such relations may have annotations:

$$risk\ mod\ rel := risk\ mod\ elem \xrightarrow{\{annotation\}} risk\ mod\ elem$$

These relations may be casual or logical connections between the elements of the risk model, and annotations may for example be conditions.

In the same manner as for the risk model we define a system model to be a set of system model elements and system model relations.

$$sys\ mod := (\{sys\ mod\ elem\}, \{sys\ mod\ rel\})$$

Note that we will not go into details of system models in this deliverable. As emphasized in the introduction and above, system modelling is dealt with in other work packages. Furthermore, our abstract syntax has no specific treatment of requirement models since the requirements should be embedded in the system model as shown in Figure 4.

A system model element consists of a system model element type and a set of attributes.

$$sys\ mod\ elem := (sys\ mod\ elem\ type, \{attribute\})$$

A system model relation is a means for relating the elements of a system model.

$$sys\ mod\ rel := sys\ mod\ elem \xrightarrow{\{annotation\}} sys\ mod\ elem$$

If we, as an example, think of the system model as a class diagram, **class** would be one system model element type and the class name would be one of its attributes. Associations between classes would be one kind of system model relation and the cardinalities of the associations would be annotations.

In our terminology, as described in Section 2, we include in a target description not only the system that is our target of analysis, but also such concepts as assets, assumptions, environment, etc. Since we assert that a system model serves as a target description, we allow system model to also include elements to represent these concepts.

As explained in the introduction, when we define modelling support for risk analysis of changing and evolving systems we are also in need of mapping models that define mappings of risk model elements to system model elements. We define a mapping model as a set of mappings:

$$map\ mod := \{map\}$$

In this report a mapping is a relation from a risk model element to a system model element. As with risk model relations and system model relations, a mapping may have a set of annotations:

$$map := risk\ mod\ elem \mapsto_{\{annotations\}} sys\ mod\ rel$$

Typically, but not exclusively, a mapping model will map the risks of the risk model to the assets of the system model. A typical annotation would be the consequence of an incident on an asset.

Below, as an example, we show a definition of fault trees using the abstract syntax. An instantiation of the abstract syntax for the CORAS language is given in Section 4.1.1.

3.1.1Example: Fault Trees

As noted above, fault trees may be characterized as a kind of risk model. In this section we define fault trees (in a simplified and incomplete version) by means of the abstract syntax defined above. An example fault tree is shown in Figure 6.

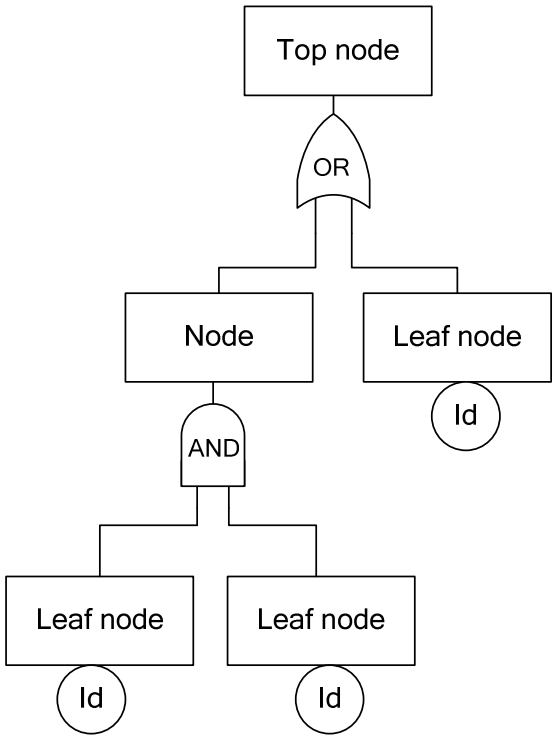

Figure 6 Example fault tree

A fault tree has three types of elements, a top node, nodes and leaf nodes:

$$risk\ mod\ elem\ type := \textbf{top node} \mid \textbf{node} \mid \textbf{leaf node}$$

All nodes have a description and a probability as attributes. Leaf nodes do in addition have an id or reference:

$$attribute := description \mid probability \mid id$$

$$top\ node := (\textbf{top node}, description, probability)$$

$$node := (\textbf{node}, description, probability)$$

$$leaf\ node := (\textbf{leaf node}, id, description, probability)$$

$$risk\ mod\ elem := top\ node \mid node \mid leaf\ node$$

We categorize the top nodes, nodes and leaf nodes into child nodes and parent nodes, which functions as the source and target of a parent relation.

$$parent\ node := top\ node \mid node$$

$$child\ node := node \mid leaf\ node$$

The relations may be annotated with **AND** or **OR**, depending on which logical gate in the fault tree they represent.

$$annotation := \textbf{AND} \mid \textbf{OR}$$

$$risk\ mod\ rel := child\ node \xrightarrow{\textbf{AND}} parent\ node \mid child\ node \xrightarrow{\textbf{OR}} parent\ node$$

If we impose some constraints on the model – for example that each child node may only have one parent and that all child nodes of a parent node must be related to the parent with the same relation (**AND** or **OR**), etc. – we have a simple definition of fault trees.

3.2 Maintenance Perspective

The maintenance perspective deals with the situation where we have a previous risk analysis that has become outdated after a certain time of system evolution, and we want to update the system documentation, the risk documentation and the risk analysis results rather than doing it all from scratch. In the following sub-sections we present the work process, the conceptual model, the language requirements, and the generic language extensions of the maintenance perspective.

3.2.1Work Process

In the maintenance perspective we assume a pervious risk analysis has been carried out and a risk model (RM) was the output of this analysis. Further we assume the existence of a system model (SM) that functioned as the target description and was the input to the analysis. The process of maintaining the risk analysis results is shown in Figure 7.

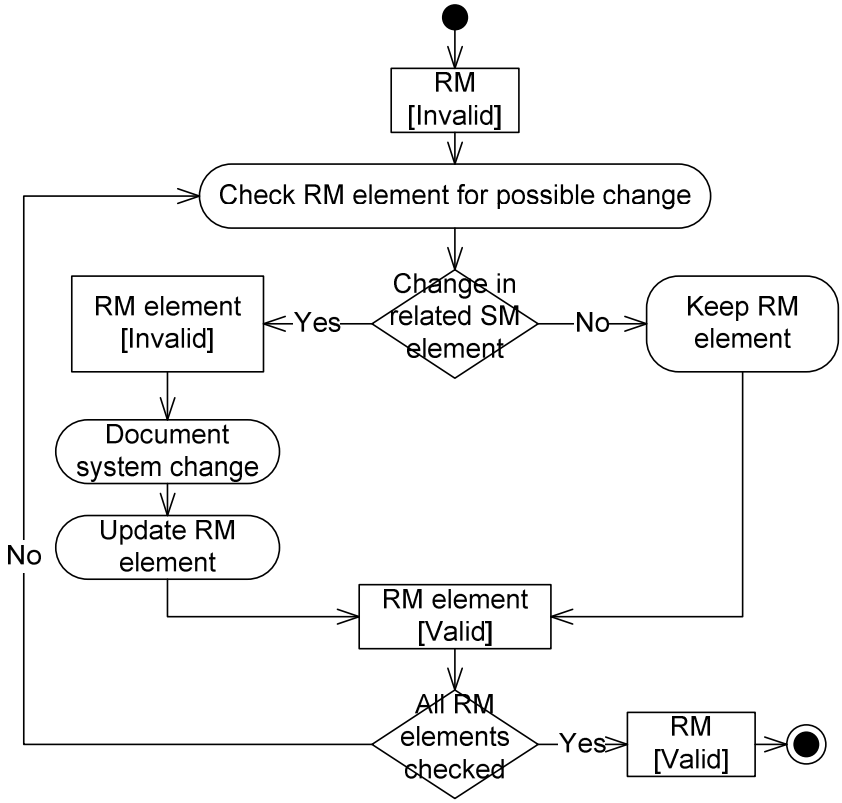

Figure 7 Process of risk analysis in the maintenance perspective

At the start of the process, we have a risk model that over time has become invalid. Each element of the risk model is checked for possible change by checking whether or not elements of the system model related to the risk model element has changed or not. In the case the related system model elements have not changed, we consider the risk model element valid and it is kept unchanged. In the opposite case, where one or more related system model elements have changed, the risk model element is considered invalid and is in need for maintenance. We then proceed to update the element(s) in the system model and then, based on the update of the system model, to update the risk model element. After all risk model elements have been checked and all invalid risk model elements are updated, the process ends with an updated risk model where the validity has been restored.

3.2.2Conceptual Model

Figure 8 presents the conceptual model of the maintenance perspective. Starting from the top we see that the perspective introduces a number of updates of the target, which means we had an old target before the updates and a current target after the updates. With both the old target and the current target there are associated risks, which we can refer to as old risks and current risks.

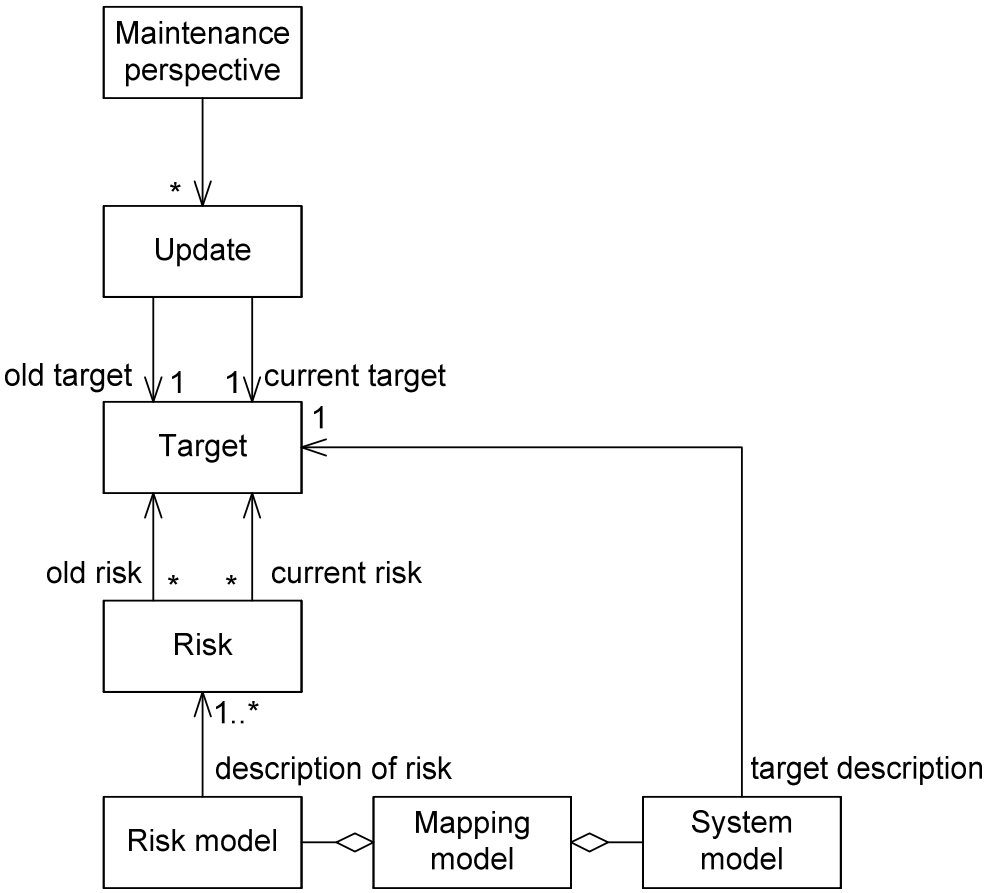

Figure 8 Maintenance perspective

In the presentation of the work process in the previous section we assumed the existence of a system model describing the before target and a risk model describing the before risks. Further, from the process it can be derived that after the maintenance analysis is completed there will be an updated system model and an updated risk model. Thus, we will also have a system model describing the after target and a risk model describing the after risks.

From the work process we can also derive the assumption of a relation between risk model elements and system model elements. In order to represent this relation, we include in the conceptual model a mapping model. The purpose of this mapping model is exactly to provide a mapping between risk model elements and system model elements. In the context of the maintenance perspective, this mapping model provides support to the task of identifying the system elements related to a given risk model element. (For an early attempt at establishing this relation, see [12]).

3.2.3Language Requirements

As should be clear from the pervious sections, maintaining risk analyses is more of a methodological problem than of a modelling language problem, and what is needed are guidelines for how to systematically update and maintain the risk models and the risk analysis results. This updating and maintenance is more of adjustments (e.g.

increase/decrease in robustness, number of attacks, number of users, required protection level, asset values, etc.) than of substantial changes of target, and likewise of the risk models. In conclusion, the language support needed for the maintenance perspective is really (a language for) the mapping model, as languages for risk models and system models already exist. This is summarized in the language requirement below.

L1. Language support for maintenance of risk analysis documentation

L1.1. Support for relating or mapping the elements of a risk model to the elements of a system model describing the target of analysis.

3.2.4Generic Extensions

The generic language extensions for the maintenance perspective should be defined so that the requirement described above is fulfilled. In the abstract syntax we define a maintenance model as a pair of snapshots, where a snapshot is a system model, a risk model and a mapping model:

$$snapshot := (sys\ mod, risk\ mod, map\ mod)$$

$$maintenance\ mod := (snapshot, snapshot)$$

The snapshots represent the old (outdated) models and the current (updated) models respectively. This means that the updates in the meta-model above are implicit in the syntax.

A risk model must have at least “risk” as an element type:[2]

$$risk\ mod\ elem\ type := \mathbf{risk} \mid \ldots$$

$$risk := (\mathbf{risk}, description, risk\ value)$$

$$risk\ mod\ elem := risk \mid \ldots$$

A system model should include the element type “asset”. In addition we assume the model has elements that describe the static and behavioural parts of the system and elements that describe relevant parts of the environment of the system.

$$sys\ mod\ elem\ type := \mathbf{asset} \mid \ldots$$

$$asset := (\mathbf{asset}, description, [asset\ value])$$

$$sys\ mod\ elem := asset \mid static\ elem \mid behavioural\ elem \mid environment\ elem \mid \ldots$$

This means we have the following attributes:

$$attribute := description \mid risk\ value \mid asset\ value \mid \ldots$$

We assume that the static, behavioural and environment elements of the system model may be related to each other, and that assets are related to static elements or behavioural elements:

$$sys\ mod\ rel := (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

[2] The **…** in the definitions below means we allow additional elements.

$$\rightarrow (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

$$\mid asset \rightarrow (static\ elem \mid behavioural\ elem)$$

Finally, the mapping model must map risk model elements to system model elements in general and risks to assets in particular:

$$map := risk \mapsto asset \mid risk\ mod\ elem \mapsto sys\ mod\ elem$$

3.3 Before-After Perspective

In the before-after perspective we are concerned with risk analyses when we anticipate major changes to the target of analysis. This means that in difference from the maintenance perspective we consider the changes before they occur rather than in retrospect. Further, it means that we are in the position where we also my consider risks that arise from the changes in the target itself. In Section 3.3.1 we present the work process of doing risk analysis in the before-after perspective, in Section 3.3.2 we define the conceptual model for the perspective, in Section 3.3.3 we formulate the requirements for language support, and in Section 3.3.4 we formalize the language extensions.

3.3.1Work Process

The work process for doing risk analysis in the before-after perspective is shown in Figure 9. Similar to the process for the maintenance perspective, we presume the existence of a system model describing the current target of analysis before the changes occur and a risk model describing the risks related to the current target. We can refer to this as the situation “as-is”. After the changes have occurred, we will have a future target with future, possibly new, risks. This we refer to the situation “to-be”. The purpose of a risk analysis in the before-after perspective is to obtain a risk model that describes the risk picture both “as-is” and “to-be”, and a risk model describing the risks associated with the transition from the target “as-is” to the target “to-be”.

We think of the changes as an operation that takes the “as-is” target as input and produced the “to-be” target. The first step of the process is to document this change operation (that of course may be the composition of a number of sub-operations). Because this change operation applies to the target, it will naturally also apply to the target description (in form of a system model). We can therefore assume that the documentation of the change operation will provide us with a system model describing the target “to-be” (or at least the means for obtaining such a system model).

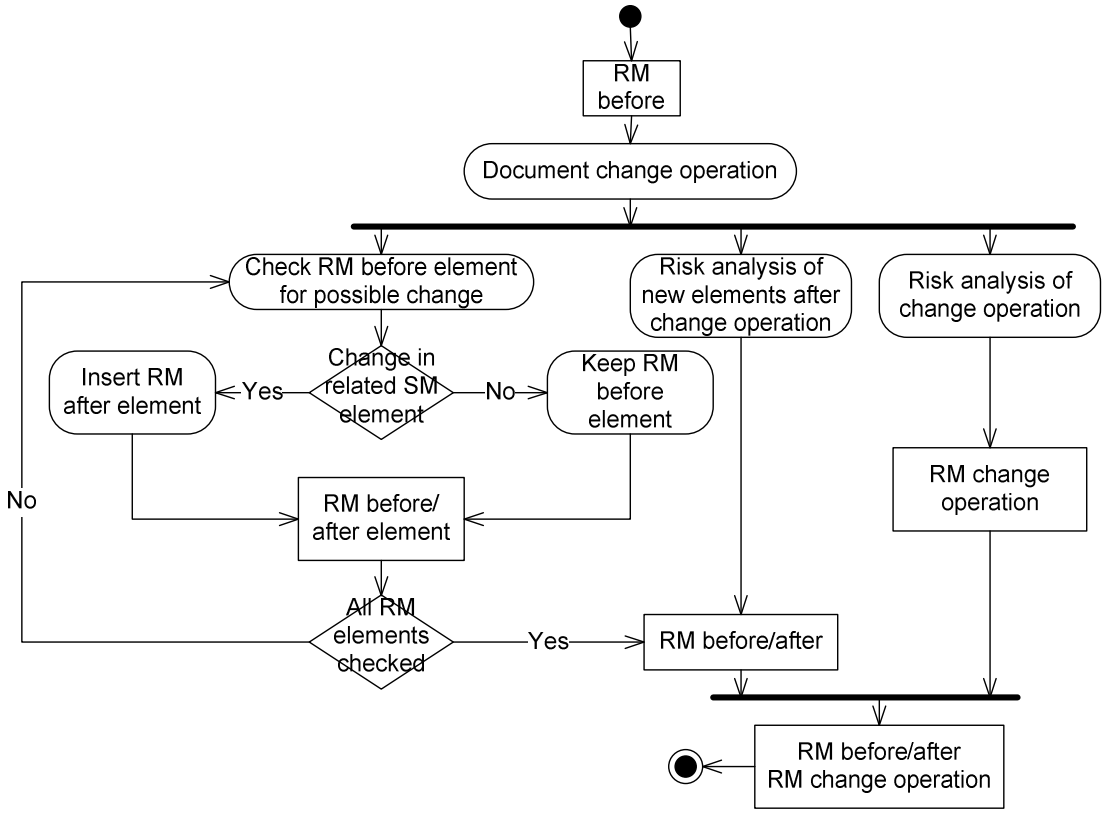

Figure 9 Process of risk analysis in the before-after perspective

After the change operation is documented, the process proceeds in three parallel tracks. The first of these parallels are similar to the process of risk analysis in the maintenance perspective: Each risk model element is checked for potential changes by checking whether or not the change operation will make changes to system model elements related to the risk model element. In case not, the risk model element is kept unchanged. In the other case, the risk model element is changed to reflect its status both before and after the change. (Note that this is different from the maintenance process where the element is simply updated to reflect the new situation.)

The second parallel of the process deals with system model elements that are new due to the change operation, i.e. elements that are present in the “to-be” system model but not in the “as-is” system model. These elements represent parts of the target of analysis that have not been analyzed with respect to risks before. A risk analysis of these elements must therefore be made and the risk related to them must be documented as risks “to-be”. In order to obtain the new risk model, that documents both the risks “as-is” and the risks “to-be”, the results of the first two parallels of the process are combined.

The third parallel of the process deals with the change operation. As explained in Section 2, the transition from the target “as-is” to the target “to-be” may in itself have risks associated with it. A risk analysis of the change operation is therefore made and the risks associated with it documented. Finally, the risk model describing the risks to the change operation is combined with the risk model describing the risks before and after the change, in order to obtain a complete risk picture of the change.

3.3.2Conceptual Model

The conceptual model for the before-after perspective is shown in Figure 10. As can be observed, it is similar to the conceptual model of the maintenance perspective, but has some essential differences. Similar to the update of the maintenance perspective, the model introduces a change operation that changes the target from one state or configuration to another and that may be perceived as functions from the current target to the future target. Risks are then associated with the current target or the future target or both.

The difference lies in that risks may also be associated with the change operation, as assumed by the work process presented in the previous section. As with risks associated with the target before and after the change, these risks are described by a risk model. In addition, the mapping model introduced in the conceptual model of the maintenance perspective includes also the change operation. The purpose of this is twofold. First, there is a need for relating risks model elements to elements (sub-operations, sub-tasks, etc.) of the change operation in the same way as risk model elements are related to system model elements. Second, this provides a means for relating elements of the system and risk model affected by the changes to the elements of the change operation that account for the changes in specific system or risk model elements.

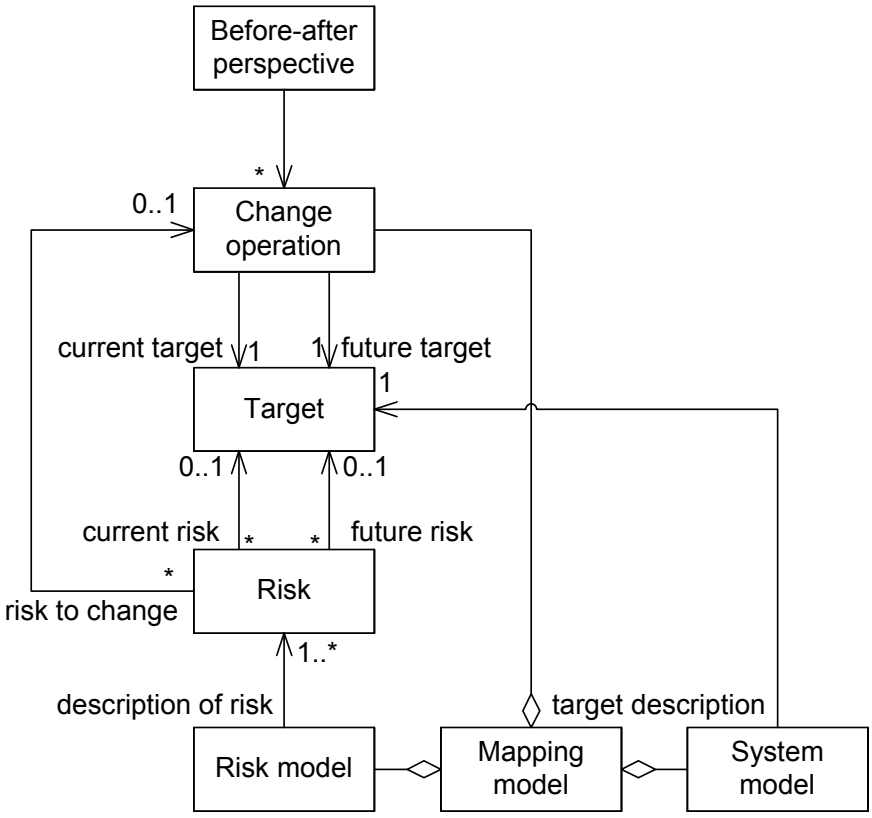

Figure 10 Before-after perspective

3.3.3Language Requirements

As is evident from the above discussion, the before-after perspective has the same requirement as the maintenance perspective that risk model elements can be mapped to system model elements. Another essential requirement is that risks both before and after a change has taken place must be represented in the risk models. In addition, language support is needed for defining change operations, relating risk model elements to the change operations and relating changes in system model elements to changes in risk model elements. These requirements are summarized below.

L2. Language support for before-after risk analysis

L2.1. Support for relating or mapping the elements of a risk model to the elements of a system model describing the target of analysis.

L2.2 Support for defining change operations.

L2.3. Support for relating risk model elements to sub-parts of a change operation.

L2.4. Support for representing risks before and after a change in risk model.

L2.5. Support for relating changes in risk and system elements to (sub-parts of) a change operation.

3.3.4Generic Extensions

In the following we define language extensions designed to fulfil the language requirements formulated above. In the abstract syntax, a before-after model is a system model, a change operation, a risk model and a mapping model.

$$\textit{before-after mod} := (\textit{sys mod}, \textit{change op}, \textit{risk mod}, \textit{map mod})$$

This means that – in difference from the maintenance model where the updates are implicit and the two versions of the models are explicit – in the before-after model, the change operation is explicit and the two versions of the models are more implicit.

The risks of the risk model should therefore have two risk values as attributes – a risk value representing the current level of risk and a risk value representing the future value of risks:

$$\textit{risk mod elem type} := \textbf{risk} \mid \ldots$$

$$\textit{risk} := (\textbf{risk}, \textit{description}, \textit{risk value}, \textit{risk value})$$

$$\textit{risk mod elem} := \textit{risk} \mid \ldots$$

In the system model we let the assets have optional asset values as attributes. As with the risks of the risk model, the assets should have asset values for the current situation and asset values for the future situation.

$$\textit{sys mod elem type} := \textbf{asset} \mid \ldots$$

$$\textit{asset} := (\textbf{asset}, \textit{description}, [\textit{asset value}, \textit{asset value}])$$

$$\textit{sys mod elem} := \textit{asset} \mid \textit{static elem} \mid \textit{behavioural elem} \mid \textit{environment elem} \mid \ldots$$

This means we have at least the following attributes:

$$attribute := description \mid risk\ value \mid asset\ value \mid \ldots$$

The system model relations are equal to the system model relations of the maintenance perspective and hold no surprises:

$$sys\ mod\ rel := (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

$$\rightarrow (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

$$\mid asset \rightarrow (static\ elem \mid behavioural\ elem)$$

As stated above, we represent the change operation explicitly in the before-after model. We define a change operation as either a basic change operation or a composite change operation:

$$change\ op := basic\ change\ op \mid composite\ change\ op$$

$$composite\ change\ op := \{change\ op\}$$

A basic change operation contains a description of the change and two system model elements – the element on which the change operation operates before and after the change:

$$basic\ change\ op := (description, sys\ mod\ elem, sys\ mod\ elem)$$

Finally, the mappings of the before-after model, in addition to mapping risks to assets and risk model elements to system model elements, also map risk model elements to change operations. This allows the model to express that there may be risks associated with the process of change itself.

$$map := risk \mapsto asset \mid risk\ mod\ elem \mapsto sys\ mod\ elem \mid risk\ mod\ elem \mapsto change\ op$$

3.4 Continuous Evolution Perspective

The continuous evolution perspective is concerned with describing risks that are associated with a target of analysis that evolves over time and that themselves evolve over time. This means that in a risk analysis in the continuous evolution perspective we make risk models that describe and forecast how the risks evolve over time. This section follows the same structure as the proceeding sections: In Section 3.4.1 we present a work process for risk analyses in the continuous perspective, in Section 3.4.2 we define a conceptual model for the perspective, in Section 3.4.3 we specify the perspective's requirements for language support, and in Section 3.4.4 we define the language extensions by means of an abstract syntax.

3.4.1Work Process

The work process for risk analyses in the continuous evolution perspective is presented in Figure 11. As with the work processes of the other two perspectives, the process assumes pre-existing (ordinary) system and risk models describing the target of analysis and the risks associated with it. For each risk model element of the risk model, the element is analysed for possible evolvement over time based on possible

evolvement over time in the related system model elements. If the analysis reveals no possibility of the system model elements, and hence the risk model element, evolving over time, the risk model element is kept as in the risk model we started with.

On the other hand, if the system model elements related to the risk model element have a potential for evolving over time, we use this information to generalize the risk model element to an element evolving over time. The result is a risk model element that has time as a parameter and in this way may be seen as a function over time.

When this process is repeated for all risk model elements, the result is the original risk model generalized with respect to evolvement over time. In other words we get a risk model with time as a parameter. We can think of this risk model as a function over time in the sense that by inserting different values for time in the model we can get forecasts of the risks associated with the target at different future points in time.

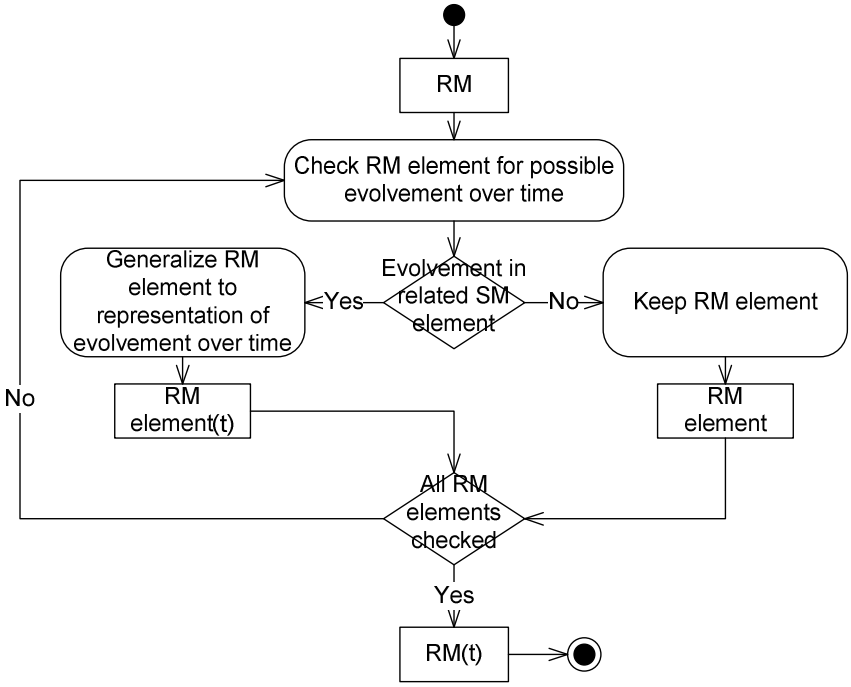

Figure 11 Process of risk analysis in the continuous evolution perspective

3.4.2Conceptual Model

The conceptual model of the continuous evolution perspective is shown in Figure 12. While the other perspectives introduce updates and change operations, this perspective introduces an evolution. An essential feature of the evolution is that it contains time as parameter. Instead of updating or changing the target as in the pervious perspective, the evolution is part of the target in the continuous evolution perspective, thus providing an evolving target. To this evolving target there are associated risks, and as can be derived from the work process presented above we can define these as evolving risks. For this reason the concept of evolution is also contained in the risks associated with the target.

As a result of this, the system model describing the target of analysis and the risk model describing the risks associated with the target should also characterize this evolution. In other words, they should have time as a parameter and be functions of time as described in the previous sections. Further, the evolution of the risk model elements should be related to the evolution of the related system model elements. In the conceptual model for the continuous evolution perspective we therefore include the evolution in the mapping model.

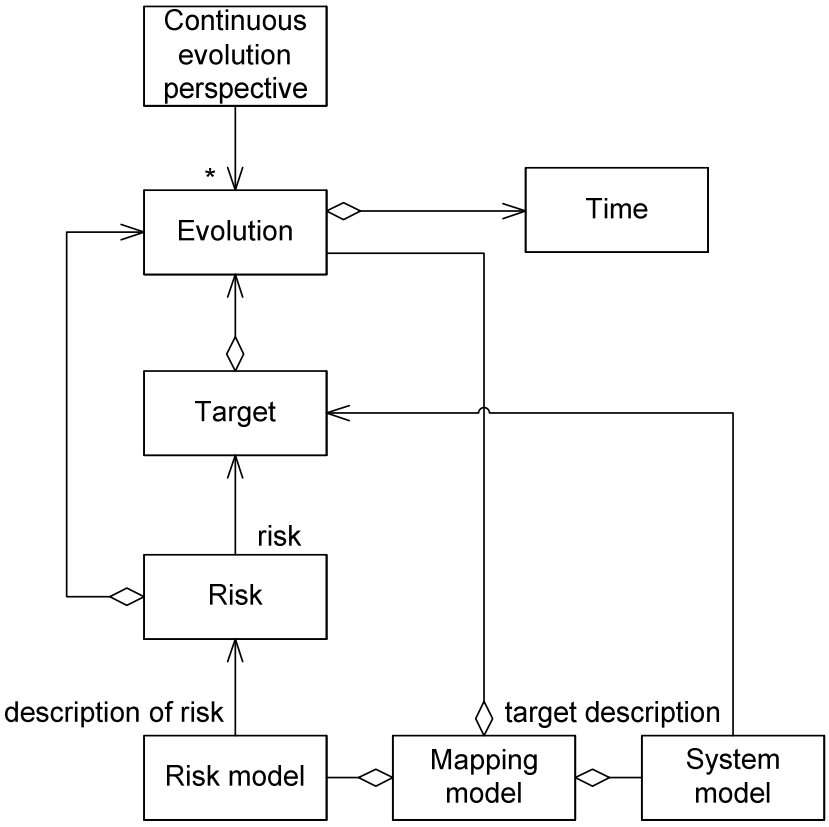

Figure 12 Continuous evolutions perspective

3.4.3Language Requirements

The crucial point in language support for the continuous evolution perspective is the possibility of representing risk model elements, and by extension risk models, as functions of time. However, as with the other two perspectives, the process of making a risk analysis with the continuous perspective also requires that we are able to relate or map the risk model elements to system model elements. The difference is that in the continuous evolution perspective we are interested in relating evolving risk model elements to evolving system model elements. In addition to this we are also interested in relating the evolution of a risk model element to the evolution of a system model element. The requirements are summarized below.

L3. Language support for risk analysis of evolving systems

L3.1. Support for representing risk model elements and risk models as functions of time.

L3.2. Support for relating or mapping evolving risk model elements to evolving system model elements.

L3.3. Support for the relating the evolution of risk models element to the evolution of system model elements.

3.4.4Generic Extensions

In the following we define the language extensions of the continuous evolution perspective by means of abstract syntax for a continuous evolution model fulfilling the requirements stated above. In the abstract syntax a continuous evolution model is a system model, a risk model, a set of evolutions and a mapping model.

$$\textit{continuous evolution mod} := (\textit{sys mod}, \textit{risk mod}, \{\textit{evolution}\}, \textit{map mod})$$

This means that, similar to the before-after perspective, the evolutions are explicitly represented in the model, while the various snapshots that may be produced from the model are implicit.

An evolution consists of a (possible formal) description and time:

$$\textit{evolution} := (\textit{description}, \textit{time})$$

The role of evolutions is to characterize how the various parts of the model evolve over time. For this reason, the risk element of the risk model has evolution as one of its attributes:

$$\textit{risk mod elem type} := \textbf{risk} \mid \ldots$$

$$\textit{risk} := (\textbf{risk}, \textit{description}, \textit{risk value}, \textit{evolution})$$

$$\textit{risk mod elem} := \textit{risk} \mid \ldots$$

The same is the case for the asset elements of the system model, even though for assets this is optional.

$$\textit{sys mod elem type} := \textbf{asset} \mid \ldots$$

$$\textit{asset} := (\textbf{asset}, \textit{description}, [\textit{asset value}, \textit{evolution}])$$

$$\textit{sys mod elem} := \textit{asset} \mid \textit{static elem} \mid \textit{behavioural elem} \mid \textit{environment elem} \mid \ldots$$

This means that we have to count evolutions among the attributes:

$$\textit{attribute} := \textit{description} \mid \textit{risk value} \mid \textit{asset value} \mid \textit{evolution} \mid \ldots$$

We also define evolution as a kind of annotation:

$$\textit{annotation} := \textit{evolution} \mid \ldots$$

The idea behind this is that when we relate the system model elements to each other we should be able to bind the evolutions of the related elements together, and we envision that this can be done by annotating the relations with evolutions in some way.

$$\textit{sys mod rel} := (\textit{static elem} \mid \textit{behavioural elem} \mid \textit{environment elem})$$

$$\xrightarrow{[\textit{evolution}]} (\textit{static elem} \mid \textit{behavioural elem} \mid \textit{environment elem})$$

$$| \; asset \xrightarrow{[evolution]} (static\ elem \mid behavioural\ elem)$$

The same idea is applied to the mappings of the mapping model. When risk model elements are mapped to system model elements and (optionally) when risks are mapped to assets, we should be able to bind the evolution of the risk model element to the evolution of the system model element. As with the system model relations we believe this can be obtained by some way of annotating the mappings with evolutions:

$$map := risk \mapsto_{[evolution]} asset \mid risk\ mod\ elem \mapsto_{evolution} sys\ mod\ elem$$

4 Instantiations of Generic Extensions for Concrete Languages

In this section we demonstrate how the language extensions defined in Section 3 can be implemented in concrete risk modelling languages. In Section 4.1 we provide instantiations in the CORAS language, in Section 4.2 we provide instantiations for Security DSML and in Section 4.3 we provide instantiations for ProSecO. The instantiations of the CORAS language are based on the abstract syntax introduced in Section 3, while the instantiations of Security DSML and ProSecO are based on the conceptual models for each of the perspectives.

4.1 Instantiations for the CORAS Language

In the following we present instantiations of the generic language extensions defined in Section 3 for the COARS language. We first provide a brief introduction to the language as it is presented in the literature, referred to as *basic CORAS*, and then provide instantiations of basic CORAS with respect to the three perspectives.

The instantiations are based on the abstract syntax introduced in Section 3. We first provide a definition of basic CORAS using the abstract syntax, and then for each of the perspectives modify this syntax to conform to the abstract syntax of the generic language extensions of the perspective. For each of the perspectives we also provide small examples of what the concrete (graphical) syntax of the CORAS language might look like after modification of the abstract syntax of basic CORAS.

As in Section 3, we keep the assumptions about the system model at a minimum. However, the CORAS method recommends UML for making target descriptions, so it might be fruitful for the reader to think of the system model as a variant of UML enhanced with concepts and constructs for security mechanisms, change and evolution.

4.1.1The CORAS Risk Modelling Language

The CORAS [4][5][6] risk modelling language has been designed to support communication, documentation and analysis of security threat and risk scenarios. It was originally defined as a UML profile, and has later been customised and refined in several aspects, based on experiences from industrial case studies, and by empirical investigations. It consists of the graphical syntax of the CORAS diagrams, and a textual syntax and semantics translating the graphical elements into English.

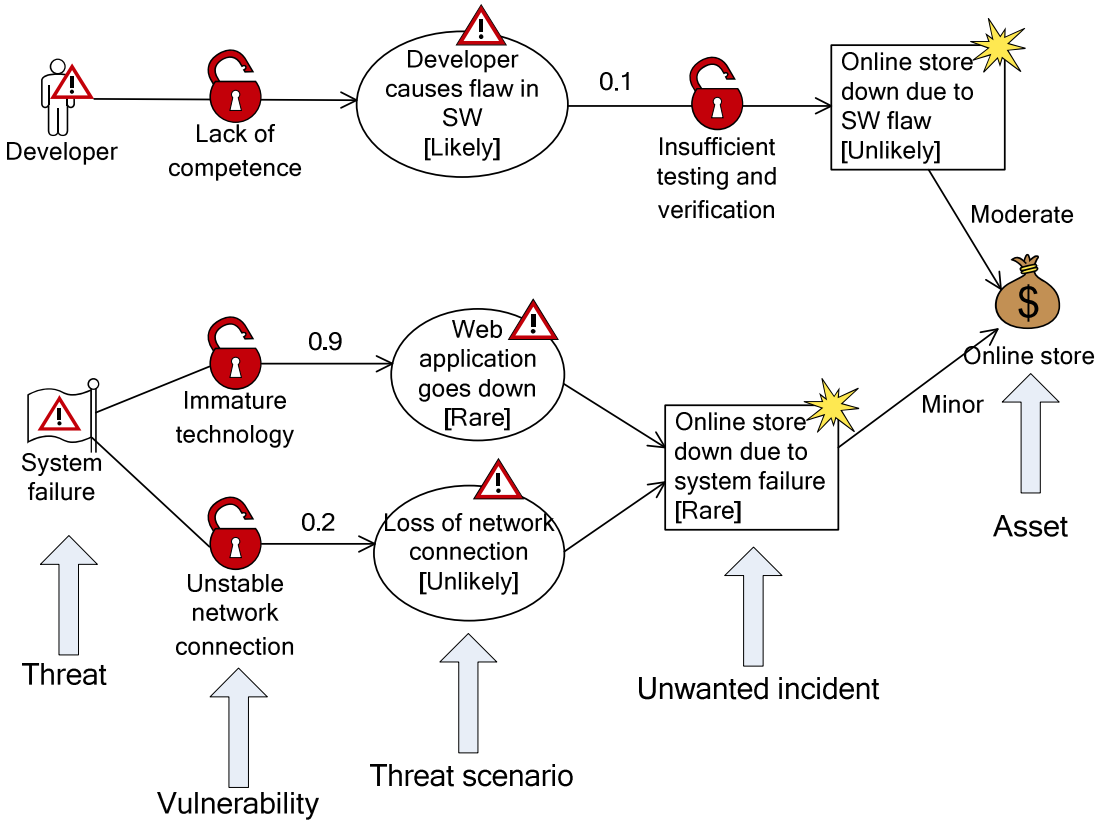

Figure 13 CORAS threat diagram

CORAS threat diagrams are used during the risk identification and estimation phases of the CORAS risk analysis process. They describe how different threats exploit vulnerabilities to initiate threat scenarios and unwanted incidents, and which assets the unwanted incidents affect. A threat diagram organizes this information in a directed acyclic graph, offering the same flexibility as cause-consequence diagrams and Bayesian networks, but using a graphical syntax that is more intuitive and easy to read. At the same time the semantics ensure that the translation of a diagram into English is unique.

CORAS diagrams were originally designed for qualitative analysis. Likelihood and consequence values are assigned directly by workshop participants during brainstorming sessions. However, the CORAS method provides a calculus [5] for computing likelihood and consequence values. The likelihood of an element may be deduced given the likelihood assigned to its parent elements and the relations leading to it. The calculus is also used to check the consistencies of assigned likelihood values.

An example of a diagram in the CORAS language is shown in Figure 13. Using the abstract syntax defined in Section 3.1 we can define a CORAS model as follows:

risk mod elem type := **threat** | **threat scenario** | **unwanted incident** | **vulnerability** | **risk**

attribute := *description* | *likelihood* | *unwanted incident* | *risk id* | *risk value* | *asset value*

risk mod elem := *threat* | *threat scenario* | *unwanted incident* | *vulnerability* | *risk*

threat := (**threat**, *description*)

$$\textit{threat scenario} := (\textbf{threat scenario}, \textit{description, likelihood})$$

$$\textit{unwanted incident} := (\textbf{unwanted incident}, \textit{description, likelihood})$$

$$\textit{vulnerability} := (\textbf{vulnerability}, \textit{description})$$

$$\textit{risk} := (\textbf{risk}, \textit{risk id, unwanted incident, risk value})$$

$$\textit{annotation} := \textit{likelihood} \mid \textit{vulnerability} \mid \textit{consequence}$$

$$\textit{risk mod rel} := \textit{threat} \xrightarrow{likelihood,\{vulnerability\}} \textit{threat scenario}$$

$$\mid \textit{threat scenario} \xrightarrow{likelihood,\{vulnerability\}} \textit{threat scenario}$$

$$\mid \textit{threat scenario} \xrightarrow{likelihood,\{vulnerability\}} \textit{unwanted incident}$$

$$\mid \textit{unwanted incident} \xrightarrow{likelihood,\{vulnerability\}} \textit{unwanted incident}$$

$$\textit{sys mode elem type} := \textbf{party} \mid \textbf{asset}$$

$$\textit{sys mod elem} := \textit{party} \mid \textit{asset}$$

$$\textit{party} := (\textbf{party}, \textit{description})$$

$$\textit{asset} := (\textbf{asset}, \textit{description}, [\textit{asset value}])$$

$$\textit{sys mod rel} := \textit{party} \rightarrow \textit{asset}$$

$$\textit{map} := \textit{unwanted incident} \mapsto_{\textit{consequence}} \textit{asset} \mid \textit{risk} \mapsto \textit{asset}$$

4.1.2Instantiations for the Maintenance Perspective

Support for risk analyses in the maintenance perspective is more about methodological support than about language support. The approach to support the maintenance perspective in CORAS is therefore to extend the CORAS method with guidelines for how to maintain results and how to document results in a way that facilitates maintenance. This means that no new modelling constructs as such are needed and we keep the basic CORAS language unchanged. However, as noted in Section 3.2, we need to support for specifying mappings between system models and risk models.

Most of the definitions in the abstract syntax of basic CORAS are unchanged, and we do not repeat them here. However, we need a somewhat more elaborate system model than assumed in Section 4.1.1, and therefore adopt the system model assumed in Section 3.2.4 (generic language extensions for the maintenance perspective).

This means first that we allow more attributes, annotations and system model elements that defined for basic CORAS. We do this by appending the … to the definitions:

$$\textit{attribute} := \textit{description} \mid \textit{likelihood} \mid \textit{unwanted incident} \mid \textit{risk id} \mid \textit{risk value} \mid \textit{asset value} \mid \ldots$$

$$\textit{annotation} := \textit{likelihood} \mid \textit{vulnerability} \mid \textit{consequence} \mid \ldots$$

$$\textit{sys mode elem type} := \textbf{party} \mid \textbf{asset} \mid \ldots$$

Second, we assume that system models may (at least) contain static, behavioural and environment elements, and update the system model relations accordingly:

$$sys\ mod\ elem := asset \mid party \mid static\ elem \mid behavioural\ elem \mid environment\ elem \mid \ldots$$

$$sys\ mod\ rel := (static\ elem \mid behavioural\ elem \mid environment\ elem) \rightarrow (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

$$\mid party \rightarrow asset$$

$$\mid asset \rightarrow (static\ elem \mid behavioural\ elem)$$

With this enhanced system model definition, we define the mappings that make up the mapping model. In addition to having mappings from risk and unwanted incidents to assets, as in basic CORAS, the new mapping model also allows the elements of a CORAS model, i.e. threats, threat scenarios, vulnerabilities, unwanted incidents and risks, to be mapped to static behavioral and environment elements.

$$map := unwanted\ incident \mapsto_{consequence} asset$$

$$\mid risk \mapsto asset$$

$$\mid threat \mapsto (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

$$\mid threat\ scenario \mapsto (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

$$\mid vulnerability \mapsto (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

$$\mid unwanted\ incident \mapsto (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

$$\mid risk \mapsto (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

Figure 14 illustrates this mapping of risk model elements to system model elements. Is should be noted that we do not propose this as a notation for making these mappings, for two reasons. First, we believe that an explicit notation for these kinds of mappings will make messy diagrams and therefore obscure rather than clarify. Second, as stated above, this is a methodological problem and not a modelling problem and we should seek a way to handle this methodologically, i.e. look for ways to identify the mapping rather than ways to represent the mappings.

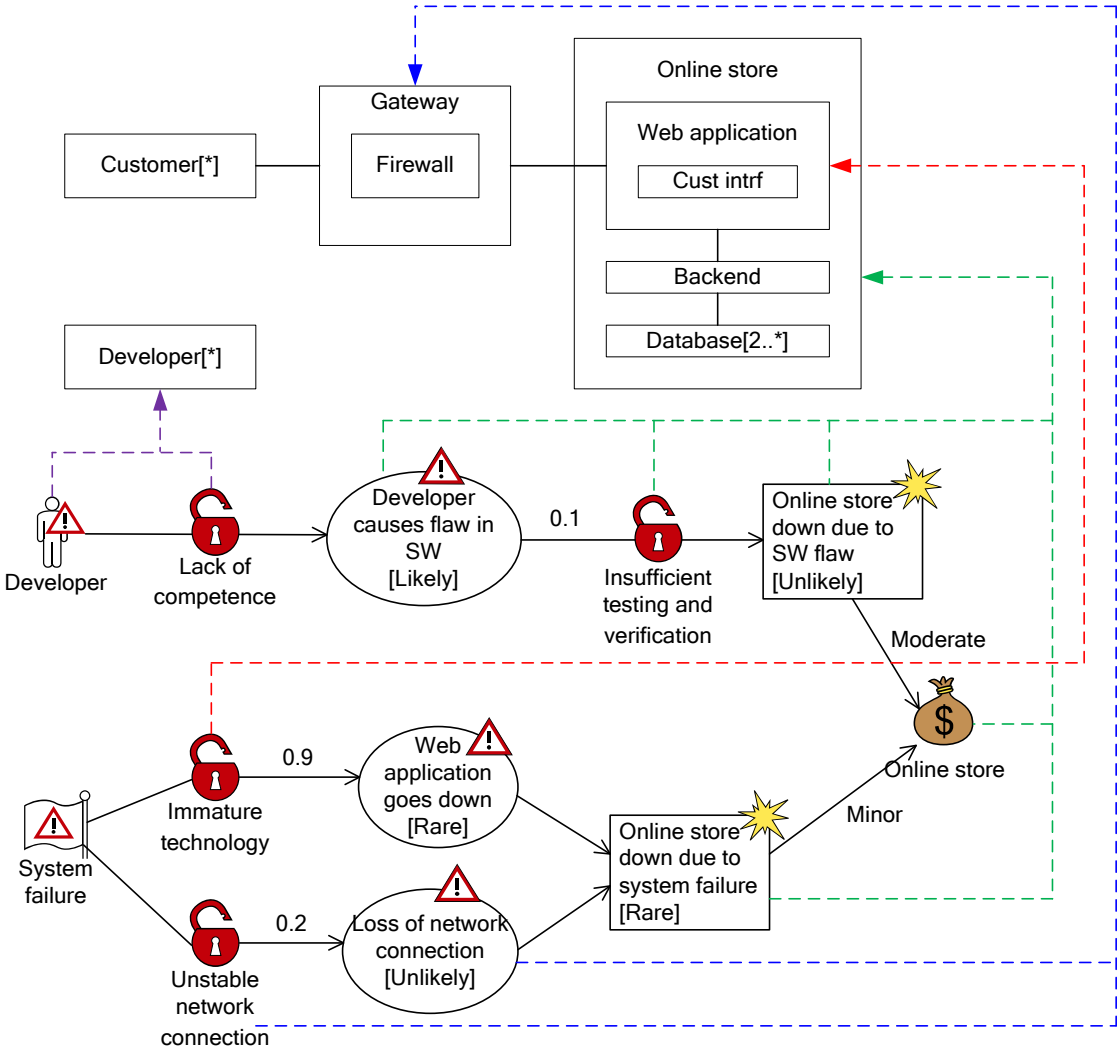

Figure 14 CORAS risk model with mappings to the system model

4.1.3Instantiations for the Before-After Perspective

In the before-after perspective we are dealing with substantial, revolutionary and planned changes. We know the target as-is and if we provide a sufficient specification of the change operation, we will also know (or be able to obtain) the target to-be. The goal of a risk analysis in the before-after perspective is to update the risk analysis *before* the changes are made, as well as analysing the risks related to the change process. The language support is therefore twofold; we need language support for documenting the changes in the risk picture resulting from the changes of the target, and we need language support for documenting the risks related to the change operations.

When instantiating the generic language extensions defined for the before-after perspective using the CORAS language – and thus extending CORAS with the necessary language support – we start by redefining the risk model elements. The risk model elements that contain likelihoods or risk values we now allow to hold two values

– one value for the current risk picture and one value for the future risk picture. The risk model elements that do not contain values in basic CORAS are allowed to hold a pair of Boolean values (bits) to represent their presence (or non-presence) in the current and in the future risk picture. These new attributes of the risk model elements are made optional for the reason that we will also map them to the change operations (change process) and then the current/future distinction does not necessarily make sense.

In the abstract syntax this is defined as follows:

$$attribute := description \mid likelihood \mid unwanted\ incident \mid risk\ id \mid risk\ value \mid asset\ value \mid \mathbf{0} \mid \mathbf{1} \mid \ldots$$

$$annotation := likelihood \mid vulnerability \mid consequence \mid \ldots$$

$$threat := (\mathbf{threat}, description, [\mathbf{0} \mid \mathbf{1}, \mathbf{0} \mid \mathbf{1}])$$

$$threat\ scenario := (\mathbf{threat\ scenario}, description, likelihood, [likelihood])$$

$$unwanted\ incident := (\mathbf{unwanted\ incident}, description, likelihood, [likelihood])$$

$$vulnerability := (\mathbf{vulnerability}, description, [\mathbf{0} \mid \mathbf{1}, \mathbf{0} \mid \mathbf{1}])$$

$$risk := (\mathbf{risk}, risk\ id, unwanted\ incident, risk\ value, [risk\ value])$$

We do the same thing for risk model relations, so that each relation has the option of being annotated with two likelihood values – one value for the current risk picture and one value for the future risk picture:

$$risk\ mod\ rel := threat \xrightarrow{likelihood,[likelihood],\{vulnerability\}} threat\ scenario$$

$$\mid threat\ scenario \xrightarrow{likelihood,[likelihood],\{vulnerability\}} threat\ scenario$$

$$\mid threat\ scenario \xrightarrow{likelihood,[likelihood],\{vulnerability\}} unwanted\ incident$$

$$\mid unwanted\ incident \xrightarrow{likelihood,[likelihood],\{vulnerability\}} unwanted\ incident$$

With respect to the system model we make the same assumptions – and hence the same definitions – as for the maintenance perspective, with the exception of the definitions of asset and party. These elements get the same treatment as the risk model elements and are allowed the additional values to represent the current and the future situation. We not repeat the definitions that are unchanged from the maintenance perspective. The new definitions of asset and party are as follows:

$$asset := (\mathbf{asset}, description, [asset\ value, asset\ value])$$

$$party := (\mathbf{party}, description, [\mathbf{0} \mid \mathbf{1}, \mathbf{0} \mid \mathbf{1}])$$

We do not make any assumptions with respect to the change operations other than what was made in the generic language extensions for the before-after perspective in Section 3.3.4, and do repeat the definitions given there. However, in the mappings of the mapping model we allow all the risk model elements to be mapped to change operations in addition to system model elements:

$map := unwanted\ incident \mapsto_{consequence,\ [consequence]} asset$

$| risk \mapsto asset$

$| threat \mapsto (static\ elem \mid behavioural\ elem \mid environment\ elem \mid change\ op)$

$| threat\ scenario \mapsto (static\ elem \mid behavioural\ elem \mid environment\ elem \mid change\ op)$

$| vulnerability \mapsto (static\ elem \mid behavioural\ elem \mid environment\ elem \mid change\ op)$

$| unwanted\ incident \mapsto (static\ elem | behavioural\ elem | environment\ elem | change\ op)$

$| risk \mapsto (static\ elem \mid behavioural\ elem \mid environment\ elem \mid change\ op)$

In addition to this, there is a second modification of the mappings. The mappings of unwanted incidents to assets may now specify two consequence values: one with respect to the current situation and one with respect to the future situation.

Figure 15 exemplifies the extensions to the CORAS language defined above with a concrete CORAS diagram. What should be noted are the pair of values in the risk model elements and on the relations, which are the main extensions of the language with respect to basic CORAS.

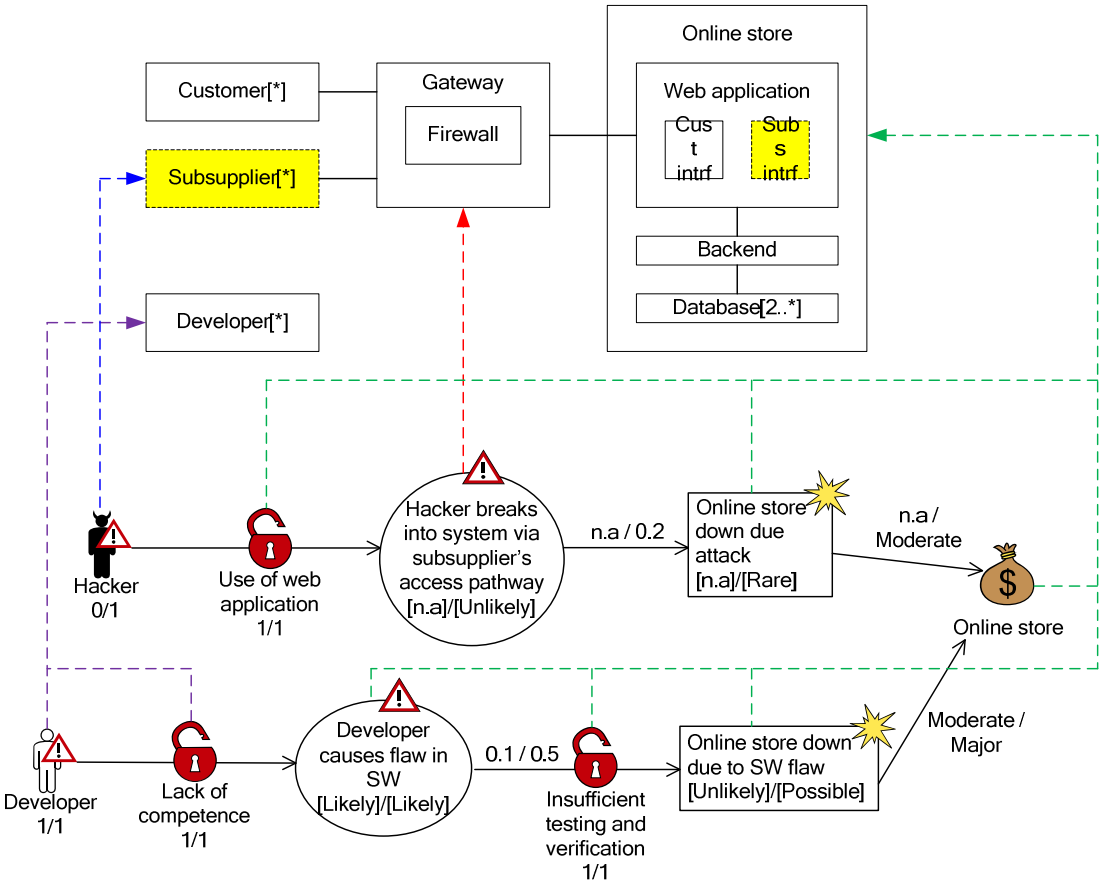

Figure 15 CORAS model in the before-after perspective

4.1.4 Instantiations for the Continuous Evolution Perspective

In the continuous evolution perspective we are concerned with the situation where we plan for the target to evolve over time or we in other ways are able to anticipate gradual changes over time. In this perspective the target description is not updated or changed as with the other perspectives; instead, evolution is a part of the target and represented in the system model. Evolution will therefore also be part of risks and we need evolution to be represented in the risk picture described in the risk model.

The challenge of defining the language extensions to the CORAS language in the continuous evolution perspective is to find a sensible way of representing the evolutions. Our solution to this challenge is to define the different values of risk model elements – i.e. asset value, likelihood, consequence and risk value – as evaluations. We also define an additional evolution called *indicator* (a concept adapted from [15]):

$$evolution := indicator \mid asset\ value \mid likelihood \mid consequence \mid risk\ value$$

An indicator is an evolution defined for system model elements other than asset and party, and is defined as a function over time or other indicators:

$$indicator := (function, time\ variable \mid \{indicator\})$$

Asset value is defined as a function over indicators:

$$asset\ value := (function, \{indicator\})$$

Likelihood is a function over indicators and other likelihoods:

$$likelihood := (function, \{likelihood \mid indicator\})$$

Consequence is a function over asset value and possibly indicators:[3]

$$consequence := (function, asset\ value, \{indicator\})$$

Risk value is defined as a function of likelihood and consequence, as is the common way of defining risk values:

$$risk\ value := (function, likelihood, consequence)$$

We can see that indirectly all the values are now defined as functions of time. As a result of this approach of defining the values of the risk model elements as evaluations, there is no need for redefining neither the risk model elements nor the risk model relations with respect to basic CORAS in order to have evolution represented in the risk model.

The same is the case for the asset element of the system models. However, as stated above, the other system model elements should contain indicators. As with the other perspective we define static, behavioural and environment elements as system elements. In difference from the other perspective we include indicators in their definitions:

$$static\ elem := (sys\ mod\ elem\ type, \{indicator\}, \{attribute\})$$

$$behavioural\ elem := (sys\ mod\ elem\ type, \{indicator\}, \{attribute\})$$

$$environment\ elem := (sys\ mod\ elem\ type, \{indicator\}, \{attribute\})$$

$$sys\ mod\ elem := asset \mid party \mid static\ elem \mid behavioural\ elem \mid environment\ elem \mid \ldots$$

When we define the system model relations we let them, with the exception of the relation from party to asset, be annotated with indicators. The reason for this is that we assume the system models have some means for specifying indicators that influence the relation between system model elements.

$$sys\ mod\ rel := (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

$$\xrightarrow{\{indicator\}} (static\ elem \mid behavioural\ elem \mid environment\ elem)$$

$$\mid party \rightarrow asset$$

$$\mid asset \xrightarrow{\{indicator\}} (static\ elem \mid behavioural\ elem)$$

We do the same thing for the mappings from risk model elements to system model elements (with the exception of the mapping from risk to asset). The rationale for this is to provide a means for specifying which indicators of a system model element that influence the risk model element mapped to it.

[3] The intuition here is that a consequence value represents loss of asset value.

$map := unwanted\ incident \mapsto_{consequence} asset$

$| risk \mapsto asset$

$| threat \mapsto (static\ elem \mid behavioural\ elem \mid environment\ elem)$

$| threat\ scenario \mapsto_{\{indicator\}} (static\ elem \mid behavioural\ elem \mid environment\ elem)$

$| vulnerability \mapsto_{\{indicator\}} (static\ elem \mid behavioural\ elem \mid environment\ elem)$

$| unwanted\ incident \mapsto_{\{indicator\}} (static\ elem \mid behavioural\ elem \mid environment\ elem)$

$| risk \mapsto_{\{indicator\}} (static\ elem \mid behavioural\ elem \mid environment\ elem)$

As a result of this, we need to define indicator as a kind of annotation:

$$annotation := likelihood \mid vulnerability \mid consequence \mid indicator \mid \ldots$$

Figure 16 shows an example of a CORAS diagram extended with the evaluations defined above. Note first that two of the system model elements define indicators with the time variable t as argument. Note also how the values of the risk model elements are expressed as functions over indicators and other values in the risk model.[4]

[4] We of course assume here that the functions f_1, f_2, f_3, f_4, and f_5 are given sensible definitions.

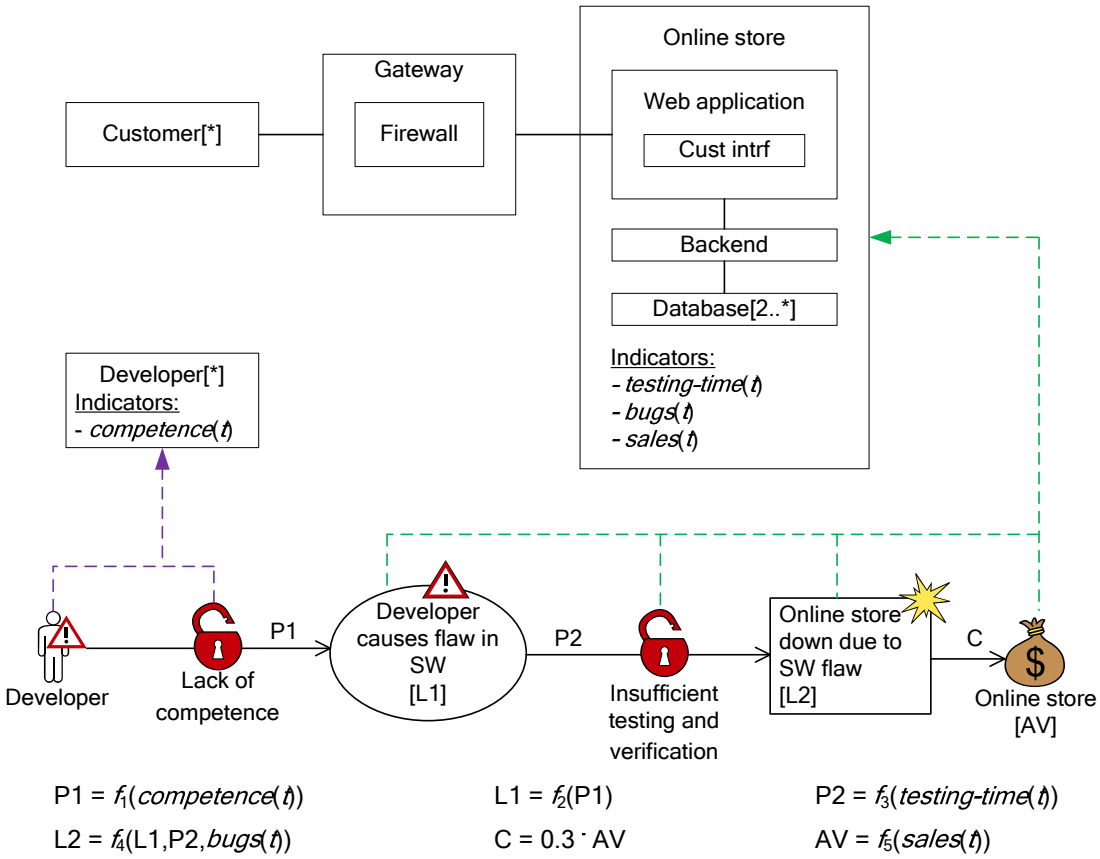

Figure 16 CORAS model in the continuous evolution perspective

4.2 Instantiations for Security DSML

As a long-term industrial initiative, Thales develops a new method to support security risk analysis, closely integrated with the overall engineering process of our critical information systems. This method is building upon model-based engineering techniques. The report [14] presents a prototype domain-specific modelling language (DSML) that was developed in this context; this DSML aims at supporting the analysis and assessment of security risks for a system, and the specification of requirements for security measures to address those risks. Our objective is to provide adequate and efficient tooling to security engineers for an effective integration of security engineering in the process of critical system design, so as to enable a better targeting of security specifications.

4.2.1Enhancing System Security Engineering in Thales

A comprehensive approach of security engineering starts with an analysis of the risks pending on the system. For critical systems, this analysis must be conducted with the biggest attention since the impacts can be very damaging for the populations in relation to those critical systems. On the other hand, overestimating risks may lead to excessive or unnecessary security measures, introducing undue rigidities and costs.

The specification of security requirements builds upon this risk analysis, and aims at defining requirements that are commensurate with the risks.

Currently in Thales, Security Analysis activities are carried out with the help of structured and proven methods that use referential repositories (of types of threats and vulnerabilities, of impacts and damages, attack scenarios, security functions etc.), standardized or not, and tabular, cross-matrix and dashboard based tools. EBIOS [7] and MEHARI [13] are the main methods employed at our company.

These methods imply a limited perception of the architecture of the system upon which the risk analysis is realised. In particular, we carried out a study of these methods and realized that they target on the one hand the business process supported by a system and on the other hand, in very little detail, technical and physical elements of the system (applications, databases, data files, servers, networks, mobile PCs etc.). Finer-grain knowledge of the architecture is not taken into account in these methods. The topology, data flows and functional dependencies throughout the system are especially not analyzed, which can lead to sub-optimal risk analyses and security requirements specifications.

Our work aims at developing a method that enables an enhancement of these classical risk analysis methodologies. As summarized in Figure 17, these enhancements rely on leveraging detailed knowledge of the targeted system in close integration with the mainstream system engineering process, and developing fine grain analyses of the actual risks at stake. This method builds upon the capacities provided by model-based development methods and techniques that are currently spreading in the systems engineering community, but are still poorly used in the security engineering domain.

Our general objectives of enhancement are the following:

- *Objective1:* To optimize the qualification of the risks and the specification of security requirements and related security costs.
- *Objective 2:* To optimize the quality and the productivity of security engineering by capitalizing on data from one study to the next, and by proceeding to automatic calculation and consistency checking.
- *Objective 3:* To optimize the quality and the productivity of security engineering by sharing common models of the system between system design and security analysis and thus by working on synchronized and consistent models of the system throughout the design process.

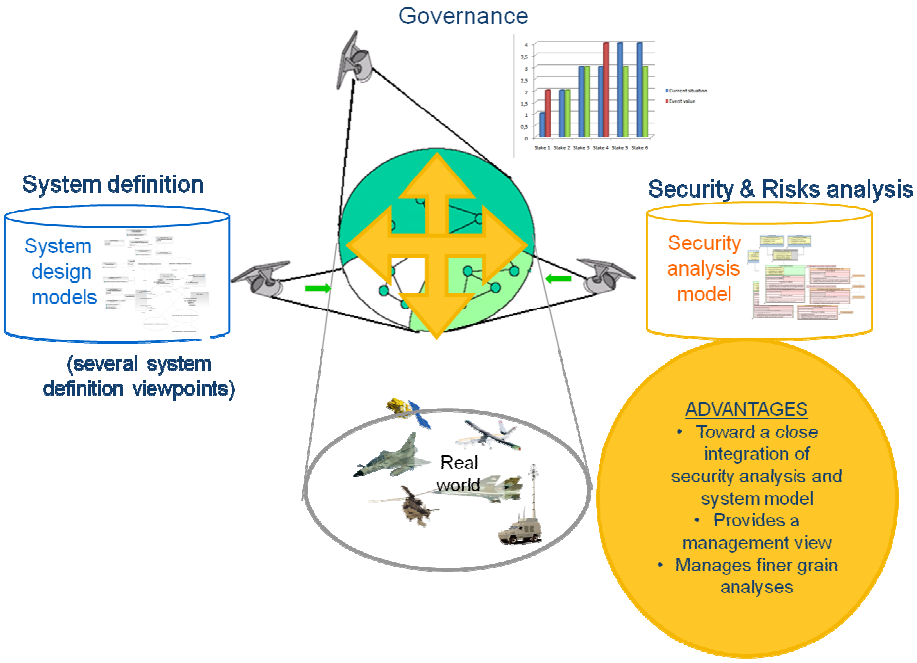

Figure 17: Enhancing system security engineering methods

4.2.2Security DSML: Overview

4.2.2.1Domain of Interest

Our goal is to build a DSML allowing the support of finer grain, more formal security analyses that exploit formalized system architecture descriptions. The security architect formalizes security information and relates it to architecture components.

Figure 18 illustrates the scope and context of use for our “Security DSML”:

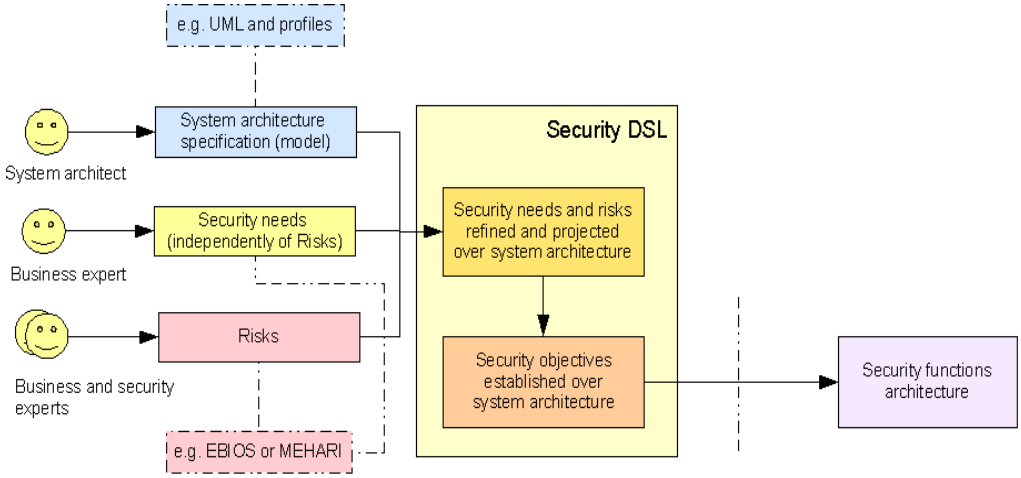

Figure 18: Scope of the Security DSML

- The System architecture model is built using distinct languages (like, for example, UML and/or UML profiles) by a System architect. *The Security DSML*

is defined with limited dependency upon the specific system architecture description formalism. The objective is to be able to use the DSML concurrently with different languages for the specification of system architectures.

- Security needs are determined for each individual architecture components or groups of such, by a Business expert. A Security need is initially expressed intrinsically (e.g. “The document needs to be defence confidential”), without taking into account the risks, but only the impacts of unwanted actions and damages they may inflict.
- A Risk Analysis takes place, involving the collaboration of the Business and Security experts, in order to identify and value risks regarding system components and subcomponents.

The Security DSML shall support security needs and risks to be refined and projected on a System architecture model.

The DSML shall then support the experts work in determining which risks are unacceptable towards the specified security needs, either because they have a too important impact, a too big opportunity of happening, or both, making these components critical.

Security objectives are defined in order to reduce unacceptable risks and consequently bring the current level of security to a newly defined targeted one. The Security DSML shall support the capture of these security objectives and the assessment of their coverage of unacceptable risks pending on architectural components.

4.2.2.2Static Model

The Risk analysis model, security requirements model and context model are expressed in a dedicated DSML. As shown by Figure 19, these kinds of models are parts of static model:

- *Requirement Model* describes the specialization of Objectives into several Requirements and the links between those and the other elements of DSML (Risk, Context).
- *Context Model* describes System Architecture (Essential Elements and/or Target), related constraints and the links between those and the other elements of DSML (Risk, Requirement).
- *Risk Model* describes the risk characterization into threats, damages and vulnerabilities and the links between those and the other elements (Requirement, Context).

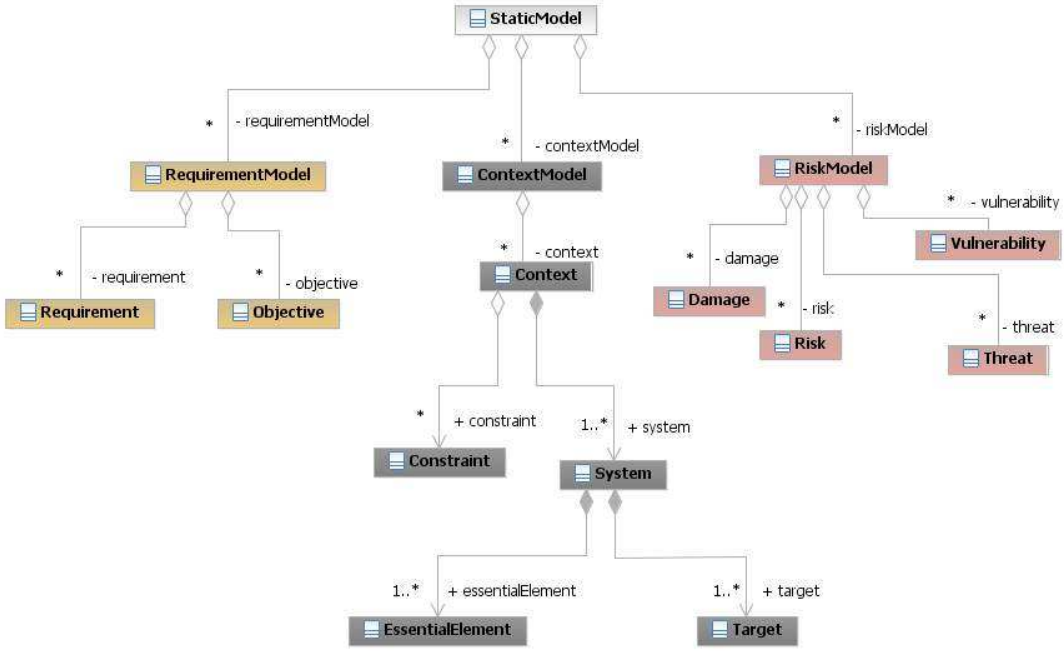

Figure 19: Security DSML Static Model description

To address change inside this DSML, we consider a change model which could be mapped with all models included in the static model. Figure 20 depicts the traceability relation between different models defined in DSML and the relation with the change model presented in section.

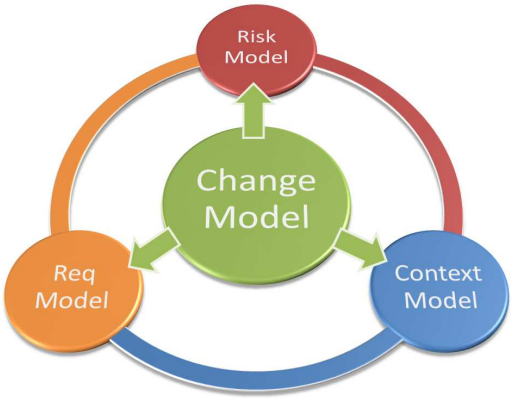

Figure 20: Relationship between DSML Static Models and Change Model

4.2.2.3Risk Model

A detailed presentation of the context meta model and syntax of Security DSML are outside the scope of this deliverable. Below we provide representative extracts; more details are provided in [14]. The core part of the Security DSML conceptual model is represented in Figure 21 below.

The system under analysis is considered to hold *targets* and *essential elements*. *Targets* are physical elements subject to *vulnerabilities* and *damages* by *threats*. *Essential elements* are usually more logical, functional elements: data and functions (or services, or capabilities depending on context) that are essential to the business stakes of the company, and therefore subject to security needs. *Essential elements* depend on *targets* for their implementation.

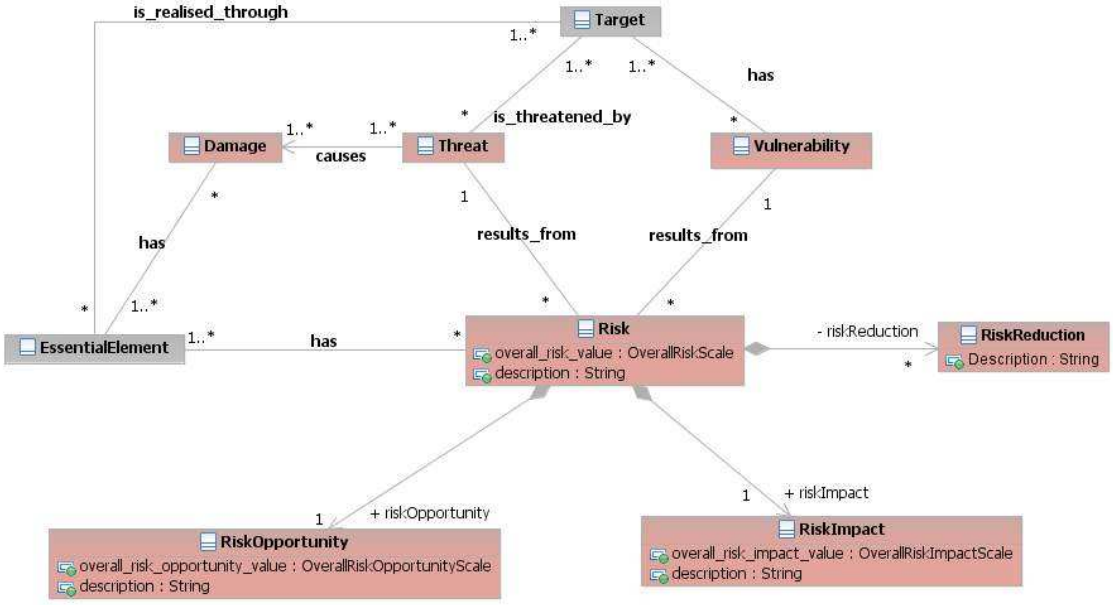

Figure 21: DSML Risk Model – Conceptual Model

The central concept of our security analysis conceptual model is the one of *risk*. A *risk* pertains to an essential element of the system. A *risk* comes from the combination of a *threat* that could exploit an opportunity to take advantage of a *vulnerability* of a target, with the *essential element* depending on the *target*. A *risk* is valuated based on its *impact* on the target and its *opportunity* to be triggered on the target. It's possible to describe by textual sentence the needed operation to *reduce Risk.*

4.2.3DSML Extension: Change Model

To represent traceability between changes and static model, we add a further Model into DSML: *Change Model* which is composed by several *Change Lines*. As shown by Figure 22, a *Change Line* is considered as set of *Changes* and *Change Transitions* to preserve links and grant consistency between successive changes which compose a Change Line.

Change is described by a Change Trigger (e.g. discover a fault or a new threat) which activates a *Change Request*. It's also possible to activate a Change Trigger by a threshold defined in an *Evolution Function* which monitors the static model of the system.

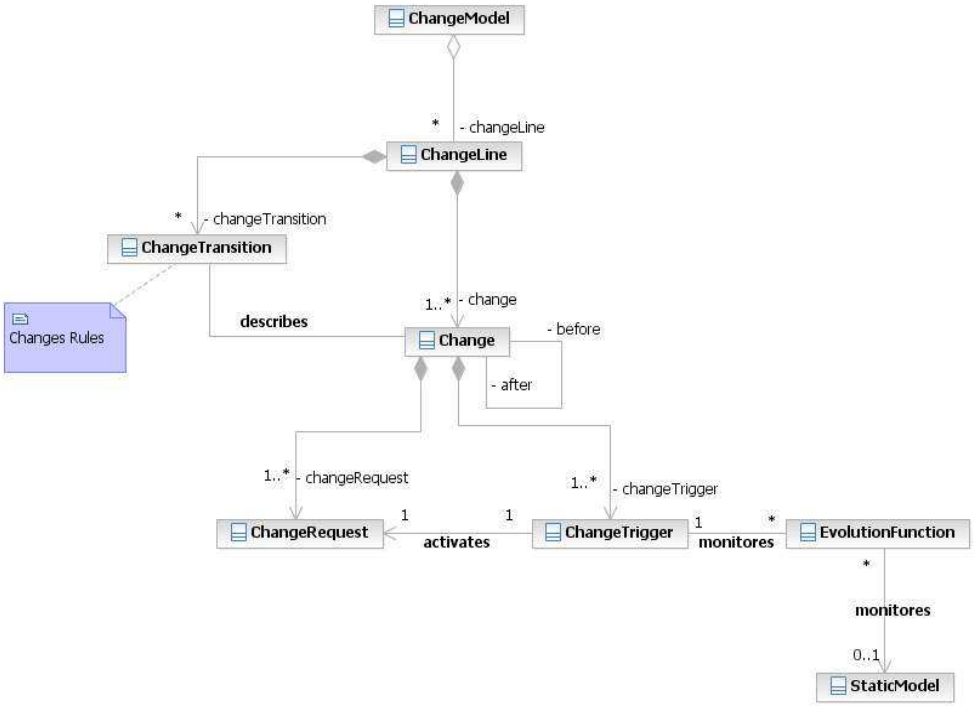

Figure 22: DSML Change Model- Conceptual Model

As shown by Figure 23, a *Change Request* contains a PUID[5] to identify it and a status which represents the state of Change request (for further detail see [16]). After the activation of Change Request by the Change Trigger, Change Request status is first defined in CCB (Configuration Control Board). The configuration (or change) control board (CCB) is a periodic meeting between several actors of a development team (client, manager, quality, design, integration …) to define change requests which are accepted, refused or postponed in the next version of system. The detailed behavior of Change Request is described in [16].

To covers all kind of static models, *Change Request* is specialized into the following kinds:

- *Requirement Change Request* modifies Requirement Model (Requirement, Objectives). It's possible to map this kind of Change Request with DOORS Change Request, for further details see [17].
- *Context Change Request* modifies Context Model (e.g. system architecture).
- *Risk Change Request* modifies Risk Model (Risk, Threat, Damage, Vulnerability).

These three kinds of Change Request are dependants; a Requirement Change Request could impact on Risk Change Request and Context Change Request and vice versa. This is why we consider a traceability relation between those Change Requests. This relation is described by an "impacts_on" association (see Figure 23).

[5] PUID = Product Unique Identifier

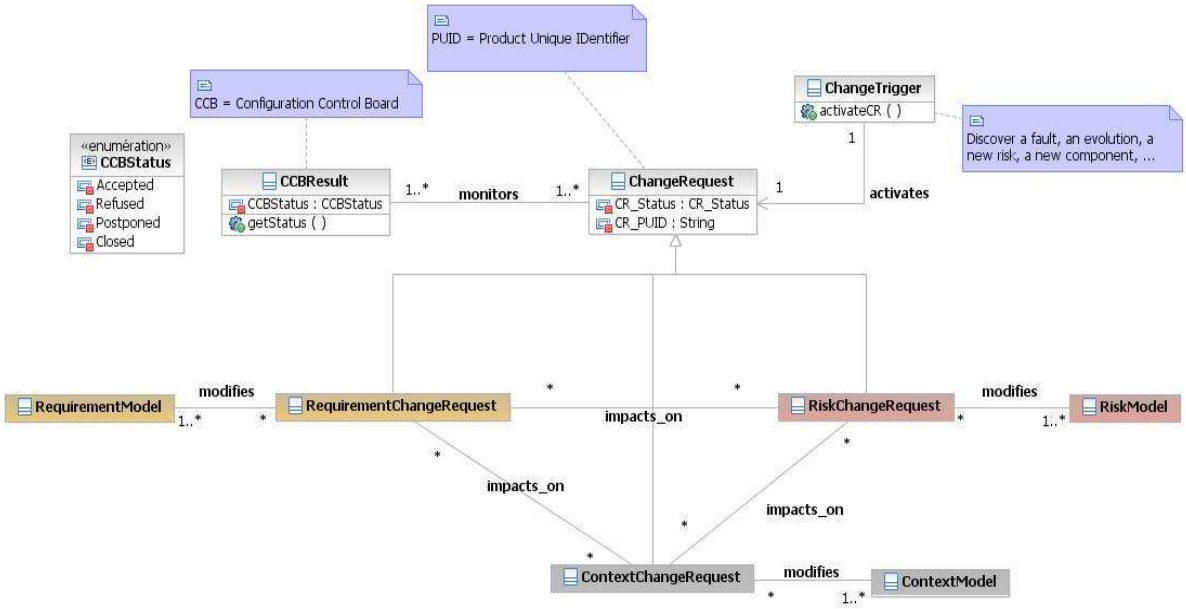

Figure 23: DSML Change Request - Conceptual Model

To define correctly a *Context Change Request*, Security Designer must first of all take into account related *constraints* of the context model. This relation is shown by association “respects” between Context Change Request and Constraint in Figure 24. These constraints describe how the service provided by the system should be realized in the context model. Independent of Platform, theses *constraints are applied on Essential Elements* of the System which describes the logical view of the system. These *constraints are evaluated on specific targets* which realize Essential Element in more concrete view dependant of the system platform. These constraints must be stored in change transition in order to preserve them on successive changes inside the change model.

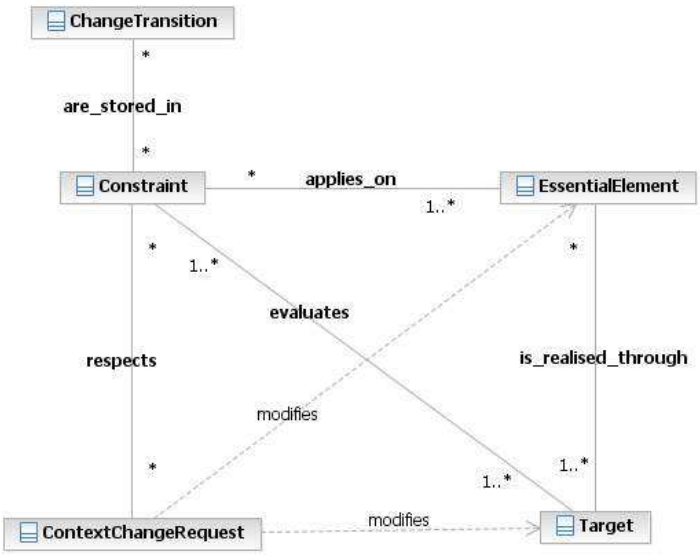

Figure 24: Relations between Context Change Request and Context Model – Conceptual Model

4.2.3.1Instantiation to the Maintenance Perspective

This sub section presents the DSML instantiation to the maintenance perspective. Starting from the top of Figure 25, we see that this perspective defines a number of updates that have some impacts in the Context Model (e.g. Essential Element, Target). This set of update activates one Change Trigger in order to open a new Context Change Request. This Context Change Request describes the updates on the Context Model and must respect the related constraint of the Context Model.

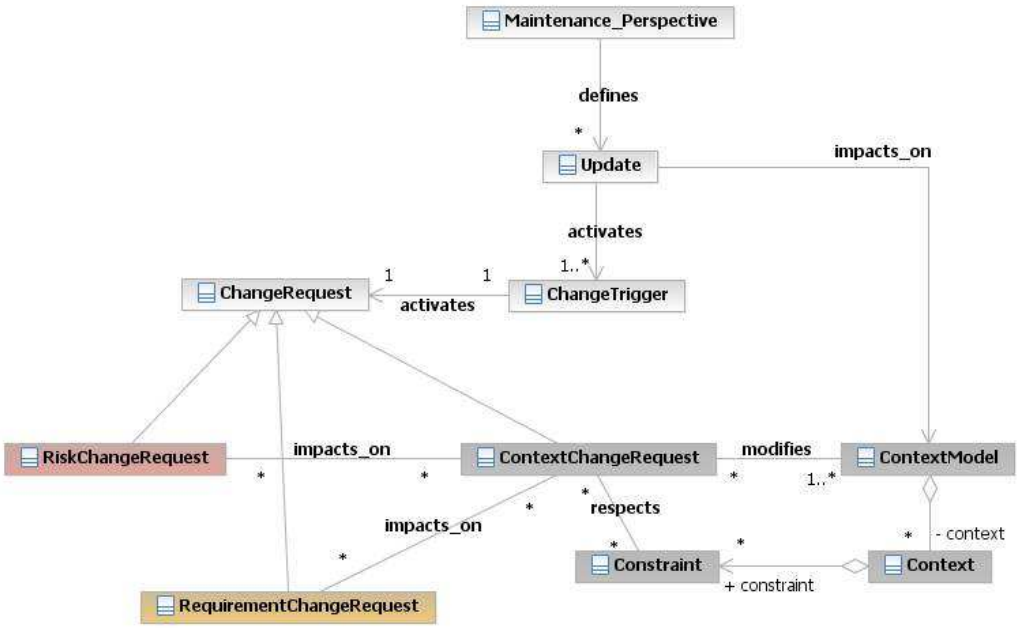

Figure 25: DSML Maintenance Perspective - Conceptuel Model

According to Figure 8, *Context Change Request* should take into account new or updated risks in the *Risk Model*. This is why *Context Change Request* could impacts on *Risk Change Request* which describes the new/updated risks on the *Risk Model* and impacts on other Change Requests, and so forth.

4.2.3.2Instantiation to the Before-After Perspective

The conceptual model of the instantiation to the before-after perspective is shown by Figure 26. This perspective defines a number of *changes* linked by a *transition* to preserve links and grant consistency between successive changes (e.g. change realized before and after). These different changes are composed by a set of *Change Requests*.

According to Figure 10, *Risks* may be associated with the *Change Request* to denote related risk on change. As in the model of change, new/updated risks are described by *Risk Change Requests* which modifies the *Risk Model* and impacts on other *Change Requests*, and so forth.

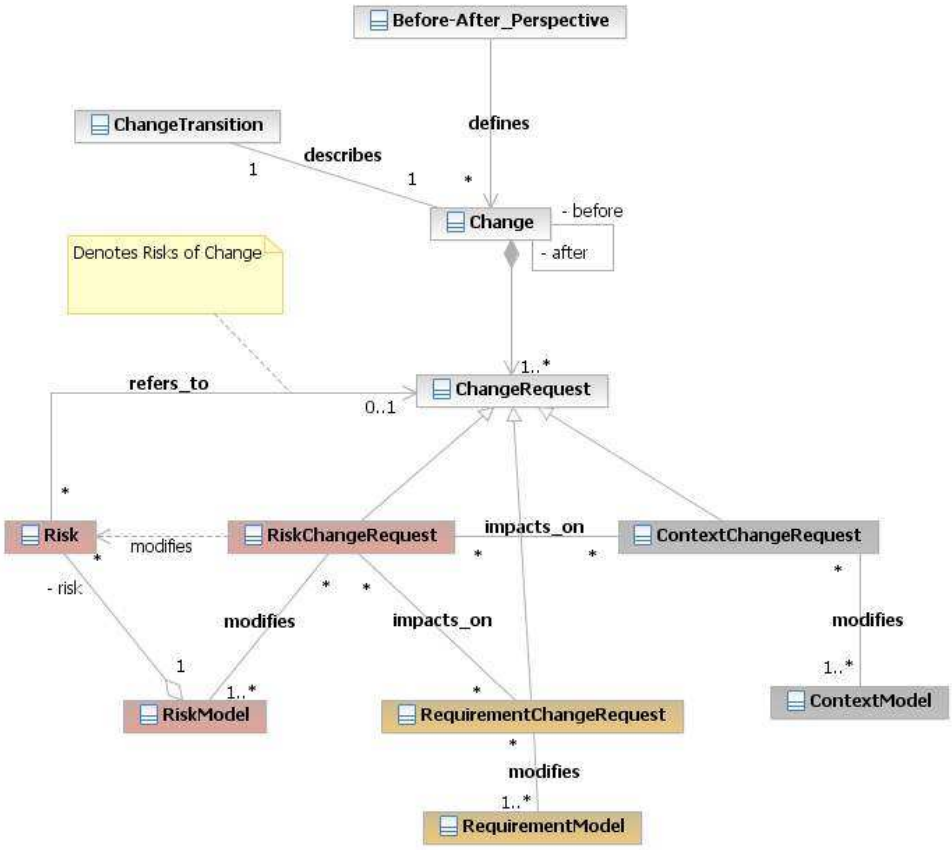

Figure 26: DSML Before-After Perspective - Conceptual Model

4.2.3.3Instantiation to the Continuous Evolution Perspective

The conceptual model of the instantiation to the continuous evolution perspective is shown by Figure 27. According to Figure 12, while the other perspectives define updates and change operations, the continuous perspective defines *evolutions functions*. An essential feature of these evolutions is that they contain *time* as parameter. In this case of perspective, *Change Trigger* is activated by a threshold defined in *Evolution Function*.

Evolution functions monitor the *static model*, as example thus providing an evolving system. To this evolving system there are associated *risks*. *Evolution functions* are also contained in the risks to the evolving system, resulting in evolving risks. As in the model of change, new/updated risks are described by *Risk Change Requests* which modifies the *Risk Model* and impacts on other Change Requests, and so forth.

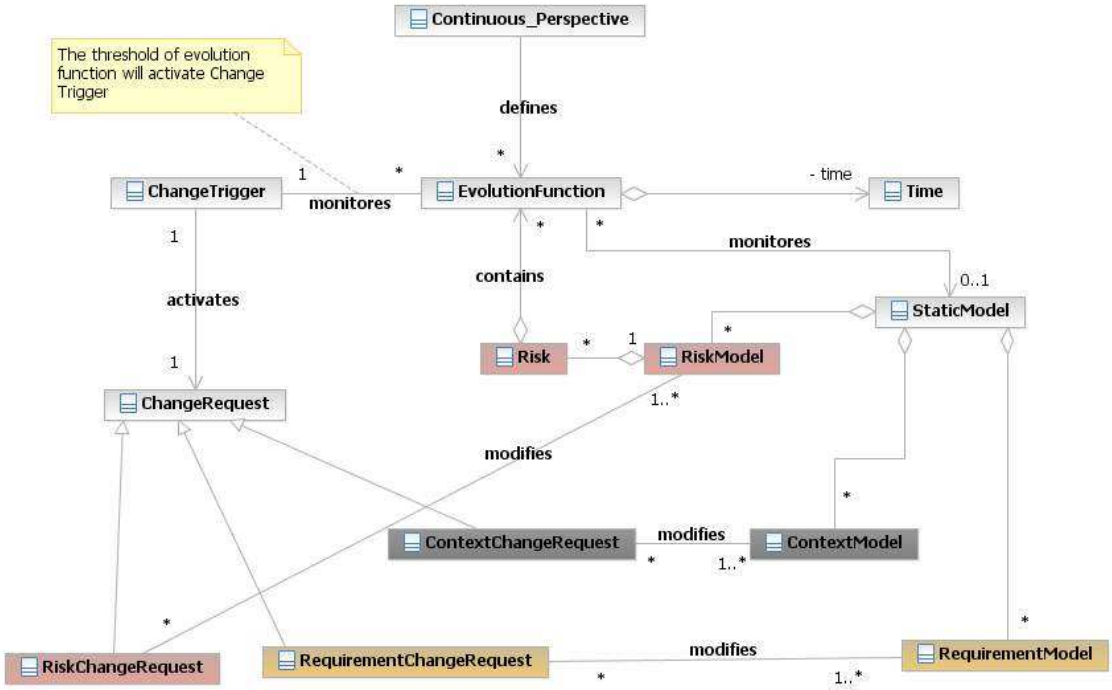

Figure 27: DSML Continuous Evolution Perspective - Conceptual Model

4.3 Instantiations for ProSecO

The ProSecO [11] approach has a specific view on system management as a process driven by change propagation. A model may be maintained by various stakeholdes in an enterprise. Changes to the system are reflected in the system model and these changes may trigger the necessity for changes of other model elements, in order to make the model consistent and to represent the changed situation correctly.

We start this section with a short introduction to the ProSecO modelling language and its view on change in general. Then we describe the specific extensions for modelling change and relate them to the meta-models given in Section 4.3.2.

4.3.1ProSecO and its View on Change

ProSecO supports different views onto a target system, modelling different aspects as e.g. the Software Engineering View or the IT-Management View. These aspects are modelled as layers (packages) in the overall System Meta Model (see Figure 28).

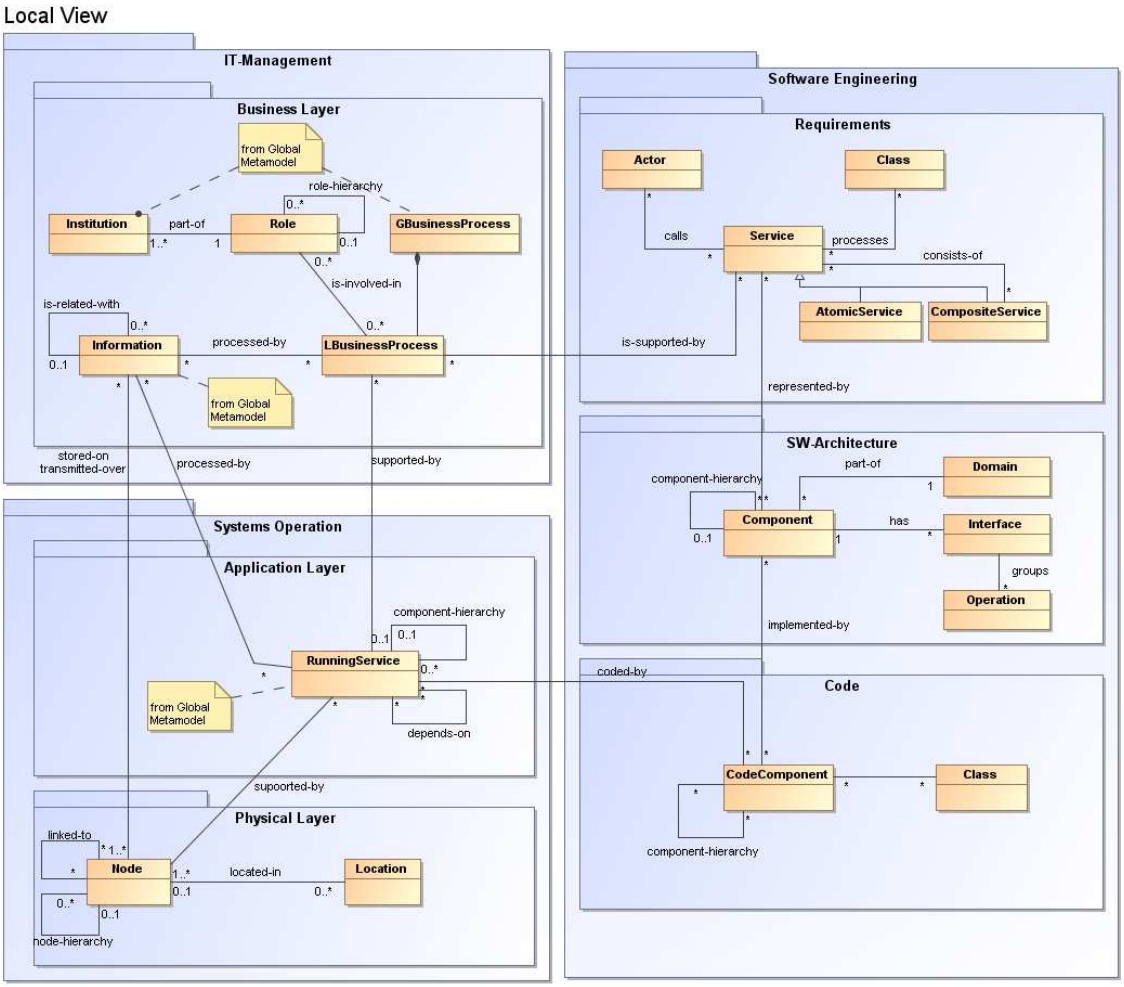

Figure 28 ProSecO Meta Model for the IT-Management, Systems Operationx and Software Engineering layer

For a complete description of the ProSecO Meta Model we refer to [16] and [9].

The security meta model is considered as an extension (called plug-in) (see Figure 29). In the centre of this plug-in is the concept of the *ModelElement* which is the super class of any conceptual element in the other views. (For the sake of readability, the inheritance relationships to all those elements are not drawn in Figure 29).

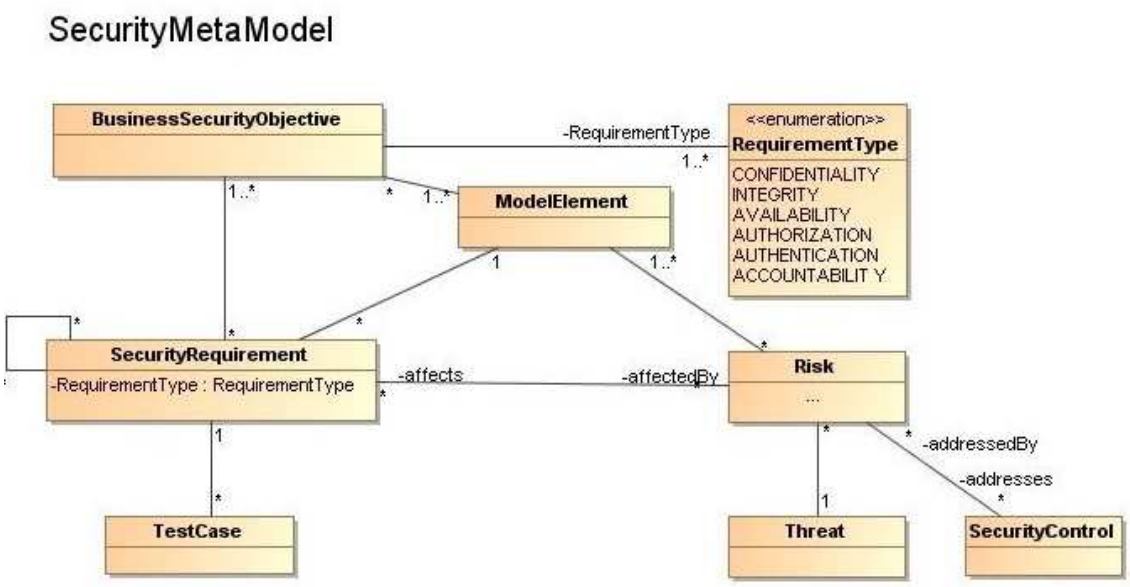

Figure 29 The ProSecO Meta Model for the Security Plugin

ModelElements are related to certain *Business Security Objectives* and/or derived *Security Requirements*, and may be subject to certain *Risks*.

4.3.1.1Change Propagation

Each element in a ProSecO model can have a specific attribute that reflects its current state. A very simple example is e.g. the state of a *Risk* as an element of the enumeration ReviewStates = {*to be reviewed, under review, controlled*}.

States are used to manage the change process in an organization. Elements in certain states may require manual interaction to act on this situation, as e.g. a *Risk* in the state *to be reviewed* may require an action of the risk manager to put a risk under review.

Furthermore each class has also an associated state machine that can react on state changes of its instances and may emit some actions, as e.g. propagating its change to other elements. In our example, this would mean, that a state change of a *ModelElement* may entail a state change in each associated *Risk* to the state *to be reviewed*. Thus state changes are propagated though the model, triggering various actions.

4.3.2Explicit Handling of Change in the ProSecO Meta Model

Besides the operational view of change in the modelling process, ProSecO is also extended to provide support to model change explicitly inside the model as a before-after perspective (and also to some extent as a maintenance perspective).

Explicit Modelling is done on the basis of the *Change Request* concept, which represents a request for a change. The change request has an explicit status out of the *ChangeRequestStatus* = {*under preparation* (the change request is discussed, but not

yet executed), *activated* (the basic change is inserted into the model), *in progress* (i.e. the consequences of the change are currently modelled), *finished* (the modelling of the consequences of the change request is finished)}[6].

When a change request is activated, the initial consequences are modelled through the concept of a *ChangeEvent*. This change event defines the initial changes defined in the change request. After that change event further state transitions of other model elements may be fired, in order to signal inconsistencies or update tasks to activate change actions by the responsible stakeholders. The status of the *ChangeRequest* is set to *in progress*.

If all *ModelElements* are in a state that correctly reflects the effects of the *ChangeRequest*, its status can be manually set to finished.

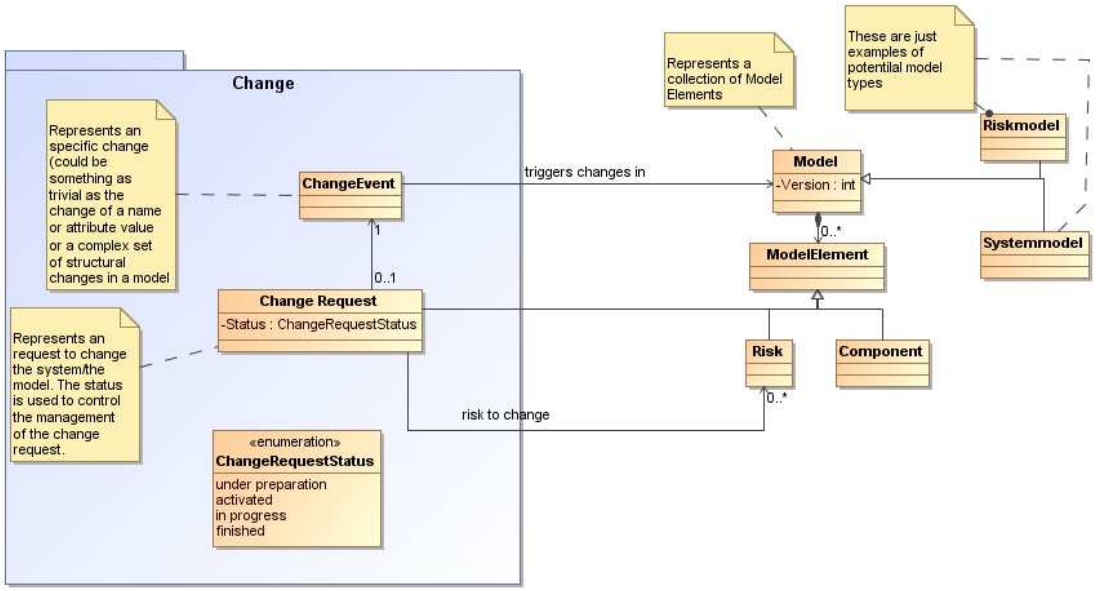

Figure 30 Explicit Handling of Change in ProSecO

4.3.2.1Before-After Perspective

The concept of the *change operation* in the meta model of the before-after perspective (Figure 10) corresponds to the *ChangeEvent* in ProSecO. The *ChangeEvent* resembles all changes to a *Model*, such it is able to represent a *current target* and an *future target* relationship in a model.

The "*risk to change"* relationship between the *Change operation* and a *Risk* corresponds to the (indirect) "*risk to change"* relationship between the *ChangeEvent* to *Risk* (via a *Change Request*).

[6] Of course, depending on the organization's internal change request process, the states and the state transitions can be far more complex.

4.3.2.2Maintenance Perspective

The concept of an update in the maintenance perspective corresponds also to a *ChangeEvent*, however without an explicit *Change Request*.

Also in the Maintenance Perspective we can derive a model before the *ChangeEvent* (although the analysis of such a model is not in the focus of the maintenance perspective).

4.3.2.3Continuous Evolution Perspective

ProSecO caters for the continuous monitoring of current system parameters. A model is able to automatically include current (and past) system parameters into its model.

The meta model can be extended by attributes that define the (planned) evolution of certain parameters. However there is no formal semantics of such an attribute.

5 Conclusions

How we handle changes in a risk analysis depends to a large degree on the context and the types of changes we are dealing with. In order to clarify the language support needed for modelling changing and evolving risk pictures we have identified three perspectives on change with respect to risk analyses:

- *The maintenance perspective*, where an old risk picture is updated after the old analysis target has been updated.
- *The before-after perspective*, where a risk picture of future changes of an analysis target is made based on a risk picture of the current analysis target.
- *The continuous evolution perspective*, where a risk picture is generalized to capture evolution of risks as the analysis target evolves.

The three perspectives require different processes and different language support. After an investigation into the perspectives, which included specifying work processes and conceptual models, we were able to formulate a number of requirements for the language support needed in each perspective. Further, we defined, for each of the perspectives, a number of generic language extensions formalized by means of an abstract syntax.

Several languages for modelling risks exist, but none with explicit support for documenting changing or evolving risk pictures. The generic language extensions formalized in the abstract syntax make only a minimum of assumptions about risk (and system) modelling languages and can therefore be implemented in a large number of risk modelling languages. By implementing the generic language extensions defined in this deliverable in a given risk modelling language – i.e. instantiating the extensions with the risk modelling language – we can obtain a risk modelling language with modelling support for the chosen perspective.

The generality and applicability of this approach are demonstrated by the instantiation of the language extensions in three different risk modelling languages: the CORAS language, Security DSML and ProSecO. In the case of the CORAS language the instantiations were made by instantiating the abstract syntax of the generic language extensions, while in the case of Security DSML and ProSecO the instantiations were made by instantiating the conceptual models.

These instantiations, in addition to demonstrating the approach, also provide three risk modelling languages with support for describing changing and evolving risk pictures.

Appendix: Glossary

Terminology is not consistently applied within the field of risk management and risk analysis. For this reason, the terminology used in the various approaches presented in this report might also be somewhat inconsistent. We have, however, strived at keeping the terminology consistent at least in the general parts of the report. In this glossary, we provide the definitions we apply, for a number of central concepts in risk analysis.

Asset: Something to which a party assigns value and hence for which the party requires protection.

Assumptions: The assumptions of the analysis are what we take as granted or accept as true (although they may not be so); the assumptions may be about the target and about the environment; the results of the analysis are valid only under these assumptions.

Consequence: The impact of an unwanted incident on an asset in terms of harm or reduced asset value.

Context: The context of the analysis is the premises for and background of the analysis; this includes the purposes of the analysis and to whom the analysis is addressed.

Direct asset: An asset that is not indirect.

Environment: The environment of the target is the surrounding things of relevance that may affect or interact with the target; in the most general case, the rest of the world.

Focus: The focus of the analysis is the main issue or central area of attention in the risk analysis; the focus is within the scope of the analysis.

Indirect asset: An asset the harm to which is completely determined by the harm to other assets with respect to the target of analysis.

Likelihood: The frequency or probability of something to occur.

Party: An organisation, company, person, group or other body on whose behalf the risk analysis is conducted.

Risk: The likelihood of an unwanted incident and its consequence for a specific asset.

Risk level: The level or value of a risk as derived from its likelihood and consequence.

Scope: The scope of the analysis is the extent or range of the target of the analysis; the scope defines the border of the analysis, i.e. what is held inside of and what is held outside of the analysis, what is the target and what is the environment.

Target: The target of the analysis is the system, organisation, enterprise, etc., or parts thereof, that is the subject of the risk analysis.

Target description: The target description is a description of the target including its focus, scope, context, environment, assumptions, parties and assets; only the parts or

aspects of the environment that are relevant for the target and the analysis are included in the target description.

Threat: A potential cause of an unwanted incident.

Threat scenario: A chain or series of events that is initiated by a threat and that may lead to an unwanted incident.

Treatment category: A general approach to treating risks; the categories are avoid, reduce consequence, reduce likelihood, transfer and retain.

Treatment scenario: The implementation, operationalization or execution of appropriate measures to reduce risk level.

Unwanted incident: An event that harms or reduces the value of an asset.

Vulnerability: A weakness, flaw or deficiency that opens for, or may be exploited by, a threat to cause harm to or reduce the value of an asset.

References

[1] ADONIS. Risk management and compliance with ADONIS: Community Edition.

[2] Asnar, Y. and Giorgini, P., Modelling risk and identifying countermeasures in organizations. In Proc. 1st International Workshop on Critical Information Infrastructures Security (CRITIS '06), number 4347 in LNCS, pages 55–66. Springer, 2006.

[3] Asnar, Y., Moretti, R., Sebastianis, M., and Zannone, N., Risk as dependability metrics for the evaluation of business solutions: A Model-driven Approach. In Proc. Third International Conference on Availability, Reliability and Security (ARES'08), pages 1240––1247, 2008.

[4] den Braber, F., Hogganvik, I., Lund, M. S., Stølen, K., and Vraalsen, F., Model-based security analysis in seven steps – a guided tour to the CORAS method. BT Technology Journal, 25(1):101–117, January 2007.

[5] Brændeland, G., Dahl, H. E. I., and Stølen, K., A modular approach to the modelling and analysis of risk scenarios with mutual dependencies. Technical report A8360, SINTEF Information and Communication Technology, 2008.

[6] Dahl, H. E. I. and Hogganvik, I., and Stølen, K., Structured semantics for the CORAS security risk modelling language. Technical report STF07 A970, SINTEF Information and Communication Technology, 2007.

[7] EBIOS – Expression of Needs and Identification of Security Objectives. http://www.ssi.gouv.fr/en/confidence/ebiospresentation.html, retrieved 23.06.2009.

[8] Giorgini, P., Mylopoulos, J., and Sebastiani, R., Goal-oriented requirements analysis and reasoning in the Tropos methodology. EAAI, 18(2):159–171, 2005.

[9] Hafner, M. and Breu, R., Security Engineering for Service oriented Architectures. Springer, 2009.

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

[11] Innerhofer-Oberperfler, F. and Breu, R., Using an enterprise architecture for IT risk management. In Proc. Information Security South Africa Conference 2006: From Insight to Foresight Conference (ISSA'06), 2006.

[12] Lund, M. S., den Braber, F., and Stølen, K., Maintaining results from security assessments. In Proc. Seventh European Conference on Software Maintenance and Reengineering (CSMR'03), pages 341–350, IEEE Computer Society, 2003.

[13] MEHARI: Information risk analysis and management methodology. https://www.clusif.asso.fr/en/production/mehari/, retrieved 23.06.2009.

[14] MODELPLEX Deliverable D3.3.g: DSML for security analysis. 2009.

[15] Refsdal, A. and Stølen, K., Employing key indicators to provide a dynamic risk picture with a notion of confidence. In Proc. IFIP Trust Management III. Third IFIP WG 11.11 International Conference (IFIPTM'09), pages 215–233, Springer, 2009.

[16] SecureChange Deliverable D2.1: An architectural blueprint and a software development process for security-critical lifelong systems. 2010.

[17] SecureChange Deliverable D3.2: Methodology for evolutionary requirements. 2010.

[18] SecureChange Deliverable D5.1: Evolution of existing methods and principles. 2009.

[19] Sindre, G. and Opdahl, A. L., Capturing security requirements through misuse cases. In Proc. 14th Norwegian Informatics Conference (NIK'01), pages 219-230, 2001.

[20] Sindre, G. and Opdahl, A. L., Eliciting security requirements by misuse cases. In Proc. TOOLS-PACIFIC, pages 120–131, 2000.

[21] Sindre, G. and Opdahl, A. L., Templates for misuse case description. In Proc. Workshop of Requirements Engineering: Foundation of Software Quality (REFSQ'01), pages 125–136, 2001.