Reuse of Security Assessment Results under Design and Maintenance of IT Systems

Folker den Braber, Mass Soldal Lund, Ketil Stølen, Fredrik Vraalsen
SINTEF Information Communication and Technology, Norway
{fbr, msl, kst, fvr}@sintef.no

Abstract
Security assessments are costly and time consuming and cannot be carried out from scratch each time a system is updated or modified. This motivates the need for specific methodology addressing the maintenance and reuse of assessment results in particular, and a component oriented approach to security assessment in general.
Traditionally, system development methodologies focus on the development of single systems. More recently, the emphasis has shifted towards the development of system product lines. We define a system product line, as a set of systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core entities in a prescribed way. A well accepted idea is that components provide the perfect foundation for the practical application of product line development. The CORAS methodology supports reuse of assessment results in a product-line oriented system development.

1. Introduction

During the EU-project CORAS (IST-2000-25031) a framework for model-based security risk assessment (in the sequel referred to as "security assessment") was developed. This framework is characterised by:

- A careful integration of aspects from partly complementary risk assessment methods like HazOp[1] [34], FTA[2] [14], FMEA[3] [5], Markov analysis [27], and CRAMM[4] [3].
- Guidelines and methodology for the use of UML[5] [30] to support the security assessment methodology.
- A security risk management process based on AS/NZS 4360 [1] and ISO/IEC 17799 [21].
- A risk documentation framework based on RM-ODP[6] [19].
- An integrated security management and system development process based on UP[7] [23].
- A platform for tool-inclusion based on XML[8] [6].

CORAS addresses security-critical systems in general, but places particular emphasis on IT security. IT security includes all aspects related to defining, achieving, and maintaining confidentiality, integrity, availability, non-repudiation, accountability, authenticity, and reliability of IT systems [20]. An IT system for CORAS is not just technology, but also the humans interacting with the technology, and all relevant aspects of the surrounding organisation and society.

1 Hazard and Operability Analysis.
2 Fault Tree Analysis.
3 Failure Mode and Effects Analysis.
4 CCTA Risk Analysis and Management Methodology.
5 Unified Modeling Language.
6 Reference Model for Open Distributed Processing.
7 Unified Process.
8 eXtensible Markup Language.

An important aspect of the CORAS project is the practical use of UML to support security management in general, and security assessment in particular. The CORAS security assessment methodology makes use of UML models for three different purposes:

- To describe the target of evaluation at the right level of abstraction. To properly assess security technical system documentation is not sufficient; a clear understanding of system usage and its role in the surrounding organisation or enterprise is just as important. UML allows these various aspects to be documented in a uniform manner.
- To facilitate communication and interaction between different groups of stakeholders involved in a security assessment. One major challenge when performing a security assessment is to establish a common understanding of the target of evaluation, threats, vulnerabilities and security risks among the stakeholders participating in the assessment. CORAS has developed a UML profile aiming to facilitate improved communication during security assessments, by making the UML diagrams easier to understand for non-experts, and at the same time preserving the well-defined ness of UML.
- To document security assessment results and the assumptions on which these results depend to support reuse and maintenance. Security assessments are costly and time consuming and should not be initiated from scratch each time we assess a new or modified system. Documenting assessments using UML supports reuse of assessment documentation, both for systems that undergo maintenance and for new systems, if similar systems have been assessed earlier.

This paper focuses on the latter purpose. In order to discuss maintenance of assessment results we need a clear understanding of how such results are documented. Section 2 is therefore devoted to a presentation of the CORAS security risk documentation framework. The various aspects of security assessment documentation are related, and precise descriptions of these relationships are essential when maintaining security assessment results. Section 3 describes such relationships in the form of dependencies. In CORAS, strategies for maintenance and reuse are documented in terms of assessment procedures. Section 4 presents examples of such procedures. Section 5 provides a summary, draws the main conclusions and describes related work.

2. Risk documentation framework

The CORAS security risk documentation framework is a specialisation of RM-ODP. It introduces the notion of “concern”. Concerns may be understood as cross-viewpoint perspectives linking together related information within the five RM-ODP viewpoints. Each concern documents results for a specific activity of the risk management process. The CORAS risk management process is divided into five sub-processes. These five sub-processes are further decomposed into activities as outlined in Table 1.

Table 1 The CORAS risk management process

Sub process 1	**Identify context**
	Activity 1.1: Identify areas of relevance
	Activity 1.2: Identify and value assets
	Activity 1.3: Identify policies and evaluation criteria
	Activity 1.4: Approval
Sub process 2	**Identify risks**
	Activity 2.1: Identify threats to assets
	Activity 2.2: Identify vulnerabilities of assets
	Activity 2.3: Document unwanted incidents
Sub process 3	**Analyse risks**
	Activity 3.1: Consequence evaluation

	Activity 3.2: Frequency evaluation
Sub process 4	**Evaluate risks**
	Activity 4.1: Determine level of risk
	Activity 4.2: Prioritise risks
	Activity 4.3: Categorise risks
	Activity 4.4: Determine interrelationships among risk themes
	Activity 4.5: Prioritise the resulting risk themes and risks
Sub process 5	**Treat risks**
	Activity 5.1: Identify treatment options
	Activity 5.2: Assess alternative treatment approaches

Each activity has one or more concerns connected to it, which are further decomposed into elements. An element may be a model, a risk assessment table, a tree, natural language text, etc. Elements may be classified into:

- Elements containing non-CORAS specific documentation, which refers to elements that are not prepared as a part of the CORAS security management process. Since CORAS should be applicable to a wide scope of systems, including already existing systems, this kind of elements is unconstrained.
- Modelling elements (constructed as part of the risk management process) specified in UML. Figure 1 presents a SWOT diagram and an asset diagram is presented in Figure 2. Both diagrams are expressed in the CORAS UML profile for security assessment [32]. The SWOT diagram illustrates graphically strengths, weaknesses, opportunities and threats, as well as stakeholders and assets on enterprise level, by stereotypes defined by the CORAS UML profile. Each strength, weakness, opportunity and threat is associated with a stakeholder and an asset. The asset diagram specifies the identified assets, their values, and which asset theme they belong to. The ownership stereotype is used to specify which stakeholder owns which assets.
- Logs from intrusion detection tools and tools for computerised vulnerability assessment.
- Risk assessment tables and trees.

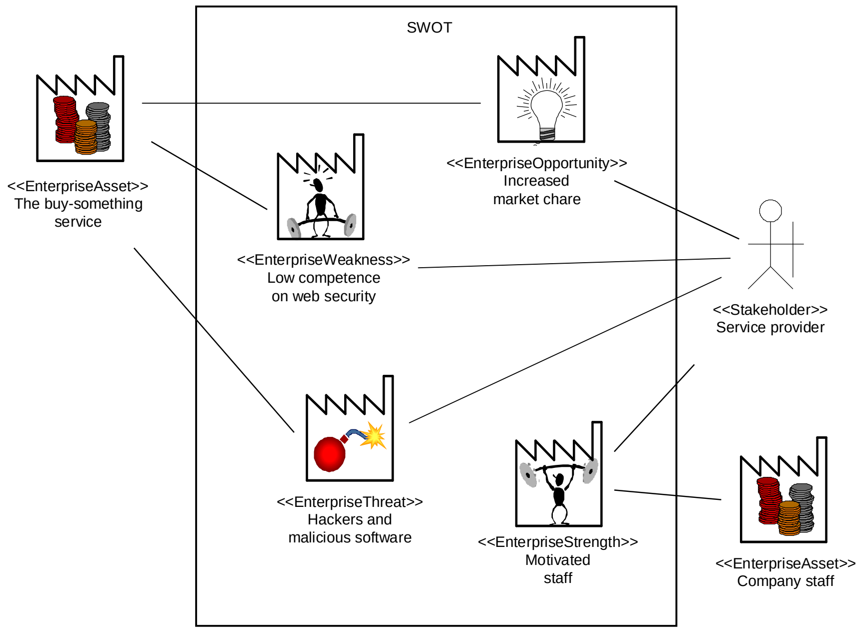

Figure 1: SWOT diagram

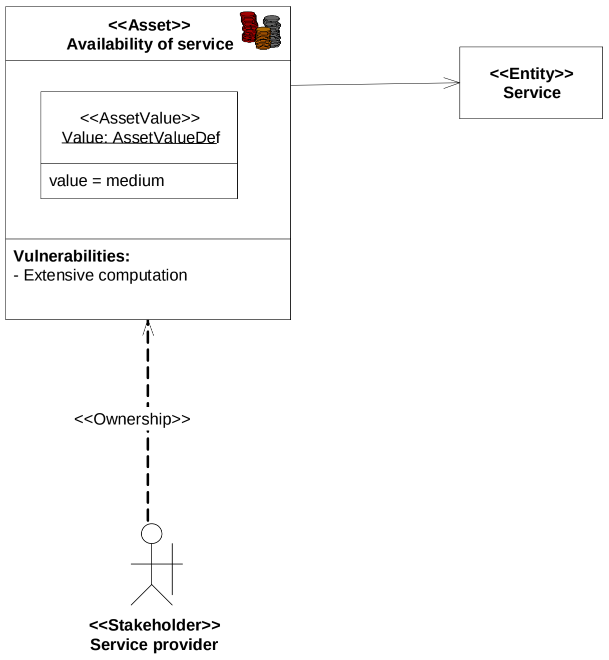

Figure 2: Asset diagram

Figure 3 presents an overview of the complete risk documentation framework with 22 concerns, five RM-ODP viewpoints, and their structuring with respect to the five security management sub-processes and their activities.

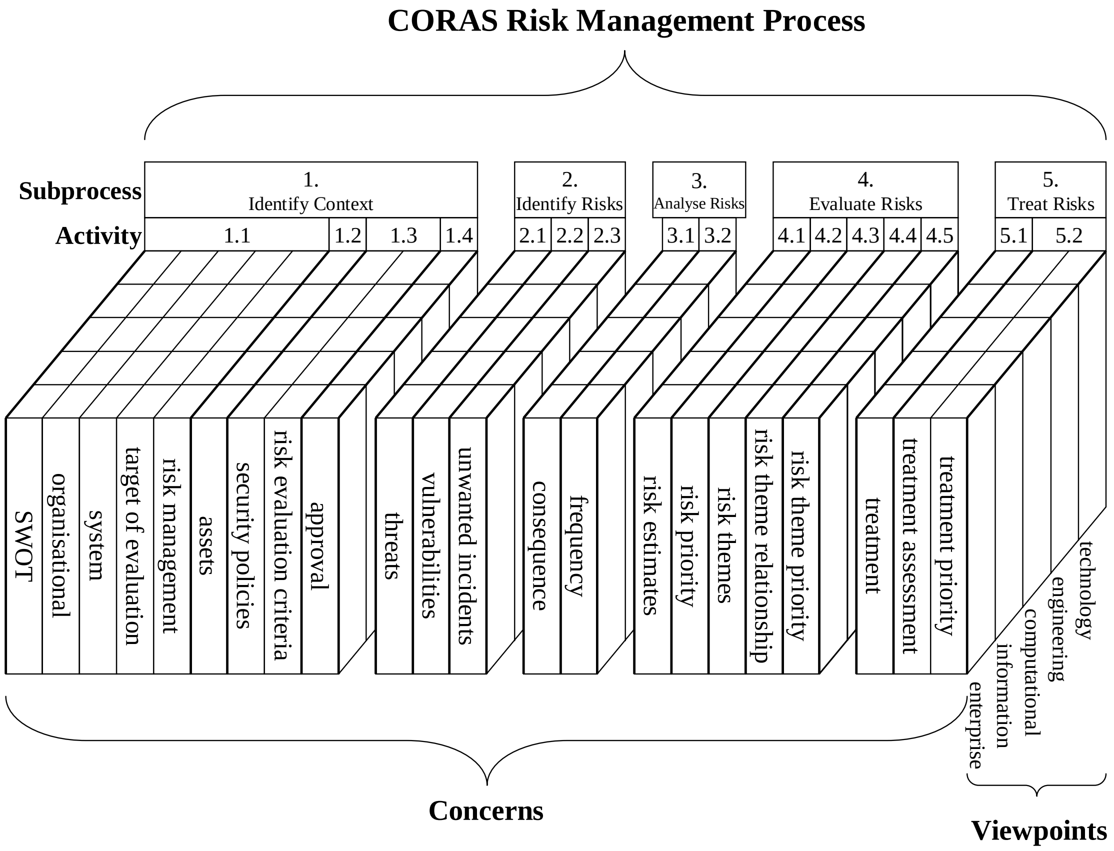

Figure 3: Risk documentation framework

3. Dependencies of assessment results

The structure of the risk documentation framework mirrors the risk management process. The data organised by the risk documentation framework have strong internal dependencies that to a large degree map to the risk management process. These dependencies also become evident in the underlying data structure of the documentation framework. For example, a vulnerability is always a *vulnerability of an asset*. This means assets must be identified before vulnerabilities; it also means that if there are no assets then there are no vulnerabilities to document. The dependencies of security assessment results are shown in Figure 4. As we will see in Section 4, these dependencies play an important role in the maintenance of risk assessment results.

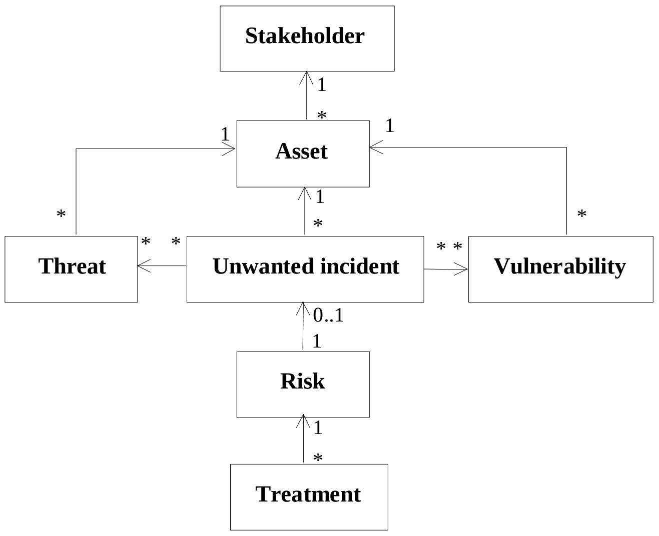

Figure 4: Dependencies of assessment results

In the setting of CORAS, the concepts of Figure 4 are given the following definitions:

- *Stakeholder.* A person or organisation who has interests in the assessed system
- *Asset.* A part or feature of the system that has value for one of the stakeholders.
- *Threat.* A potential cause of an unwanted event, which may results in harm to a system or organisation and its assets.
- *Vulnerability.* A weakness of an asset or group of assets which can be exploited by one or more threats
- *Unwanted Incident.* An undesired event that may reduce the value of an asset.
- *Risk.* An unwanted incident that has been assigned a consequence value, a frequency values, and a resulting risk value.
- *Treatment.* A way of reducing the risk value of a risk or risk theme.

As we see from Figure 4, each asset may only be related to one stakeholder. Each asset should have an unambiguous value assigned by one stakeholder. If two stakeholders view the same entity as an asset, the entity should be documented as two different assets. Two assets are per definition different if valued by different stakeholders. Both the values and the *reasons* for the valuing may be different. This means that the threats and vulnerabilities of the assets are different in the general case, and the assets should be assessed separately.

We also see that there is a many-to-one relation between unwanted incident and asset and a one-to-one relation between risk and unwanted incident. Since a risk may reduce the value of an asset, a risk belongs to one particular asset. As with stakeholders and assets, we say that if an unwanted incident may be related to two assets, we document this as two different unwanted incidents, which lead to two different risks. The same argument also holds for the relation between treatment and risk.

This enforcement of many-to-one relations provide the security assessment results with a kind of modularity that makes maintenance of the documentation much easier than it otherwise would have been. To illustrate this, imagine (the simple case of maintaining assessment results) that the assessed system is changed so that one of the assets is no longer relevant. Then all threats, vulnerabilities, unwanted incidents, risks and treatments related to that asset may be removed from the documentation without complications. In the more complicated case where the properties of an asset have changed, we are able to reassess exactly the part of the security assessment results that relates to that asset, without being concerned about how these changes may affect the results from assessing other assets.

3.1 Relationship between components and assets

The CORAS risk management process and documentation framework is *asset*-oriented. By "asset-oriented" we mean that the assets identified in Activity 1.2 guides the whole risk assessment process from Activity 1.3 onwards. A risk is always a risk of an asset, which means changes in the list of assets will lead to changes in the list of risks.

In order to develop a component-oriented methodology for maintenance, composition and reuse of assessment results, it is important to establish the relationship between components and assets, and – in order to facilitate this – to ensure that the concepts of *component* and *asset* are satisfactory defined. The definition of the latter has already been given above.

A component is an autonomous part of a system that has a well-defined border, carries out specified tasks and communicates through a well-understood interface. A component may itself consist of sub-components that may be viewed as independent components on their own. Components are not (only) "physical components" in the sense of contemporary technologies like CORBA, .NET/COM+, J2EE/EJB, but rather "logical components" that represent the logical building blocks of a system (as for example in Catalysis [12] and KobrA [3]). Such logical building blocks may contain descriptions of human behaviour; for example, in the form of work-processes.

When relating components to assets, it is useful to categorise components. We distinguish between the categories given in Table 2.

Table 2 Categories of components

software components	software or part of software
physical components	computer hardware and other physical equipment
embedded components	which has both software components and physical components as sub-components
organisational components	that may consist of software components and physical components as well as specified processes, including human behaviour

In CORAS assets are categorised into six *asset themes*, which are listed in Table 3.

Table 3 Categories of asset themes

software assets	all software used in the system or system dependent software
physical assets	all physical components in the system and system dependent physical components
information assets	all information in the system and system dependent information
human assets	assets related to human resources, specially knowledge
organisational assets	organisational concerns, organisational (system) internal regulations, routines etc.
law and regulation assets	external laws and regulations that influence the system

To describe the relationship between components and assets with respect to changes we use the relations *may affect* and *affects*. Given a target T of evaluation with assessment documentation AD formalised in accordance with the recommendations of the CORAS risk documentation framework. Changes to a component C *may affect* an asset A if the changes in C may lead to changes to AD with respect to the asset A. The changes in C *affects* an asset A if

- the value assigned to A by its stakeholder is changed;
- A is a new asset of T resulting from the change;
- A is no longer an asset of T (no longer relevant or new value equals zero);
- the list of vulnerabilities with respect to A is changed;

- new threats of relevance for *A* is introduced;
- the consequence or frequency of a risk related to vulnerability with respect to *A* is changed.

There is an important distinction between the relations *C affects A* and *C may affect A*. The latter is related to identifying assets that must be analysed, while the former is related to identifying assets for which the security assessment documentation must be updated. Clearly the former is a subset of the latter, and restricting the set of assets that may be affected to the set of assets that is affected is an important part of the maintenance process.

If a component *C* coincide with the target of evaluation *T*, it is evident that changes to *C may affect* all the assets of *T*. The way a component affects an asset may however be more intricate. We therefore define guidelines for how to identify the assets that may be affected by changes in or of a component.

It is clear that *may affect* is no one-to-one relation between components and assets and that its cardinalities are different for different kinds of assets. Assets that may be affected (i.e. defining the relation) are identified in accordance with the classification of assets and components. For example:

Software assets	A software component *C* may affect a software asset *A* if *C* is *A* or if *C* is a subcomponent of *A*
	A physical component *C* may affect a software asset *A* if *C* is hardware on which *A* runs
	An embedded component is treated as both a software component and a physical component
Physical assets	A software component *C* may affect a physical asset *A* if *C* is software that runs on *A*
	A physical component *C* may affect a physical asset *A* if *C* is *A* or *C* is a subcomponent of *A*
	An embedded component is treated as both a software component and a physical component
Information assets	A software component *C* may affect an information asset *A* if *C* is part of software used for storing or processing *A*
	A physical component *C* may affect an information asset *A* if *C* is part of hardware used for storing *A*
	An embedded component is treated as both a software component and a physical component
Organisational assets	A software component *C* may affect an organisational asset *A* if *A* regulates the use of *C*
	A physical component *C* may affect an organisational asset *A* if *A* regulates the use of *C*
	An embedded component is treated as both a software component and a physical component
	An organisational component *C* may affect an organisational asset *A* if *A* regulates the processes of *C*
Law and regulation assets	A software component *C* may affect a law and regulation asset *A* if *C* is part of processes that are regulated by *A* or if *C* is part of storing or processing of information that is regulated by *A*
	A physical component *C* may affect a law and regulation asset *A* if *C* is part of processes that are regulated by *A* or if *C* is part of storing information that is regulated by *A*
	An embedded component is treated as both a software component and a physical component

4. Procedures for maintenance and reuse

The CORAS security management process has an assumption/guarantee flavour. The security management sub-process “Identify Context” aims at characterising the overall context and scope for the assessment. The concerns documenting the results of this sub-process describe the pre-conditions on which the assessment is based.
The risk management sub-processes “Identify Risks”, “Analyse Risks”, “Evaluate Risks” and “Treat Risks” aim at a risk assessment within the context and scope captured by the assessment assumption. The concerns documenting the results from these four sub-processes describe the assessment guarantee. Hence, with respect to Figure 3, the nine concerns listed under “Identify Context” record the assessment assumption, while the remaining thirteen concerns capture the assessment guarantee.
The pair of assessment assumption and assessment guarantee may be understood as an assessment contract. An assessment contract for a component consists of an assessment assumption describing the target of evaluation as well as other pre-conditions on which the assessment builds, and an assessment guarantee describing assessment results for the component in question with respect to the assessment assumption.

4.1 Maintaining assessment results

IT systems are updated or modified on a regular basis. Connected to such updates or modifications it is often necessary to reassess their security since changes may have introduced new risks. In this section we outline a general procedure for maintaining assessment results when the assessed system undergoes maintenance. We also refine this procedure into specialised procedures for three special cases:

- component modification;
- component substitution;
- component introduction.

The procedure is defined for changes in or of *one* component, but may easily be generalised to apply for more than one component by iterating the whole procedure for each component, or by iterating each of the activities for each component.
The procedure is built around the contents of three sets:

- PAA: the set of *possibly affected assets*;
- AA: the set of *affected assets*;
- AR: the set of *affected risks*.

The procedure is outlined in Figure 5 (*C* refers to the component to be maintained).

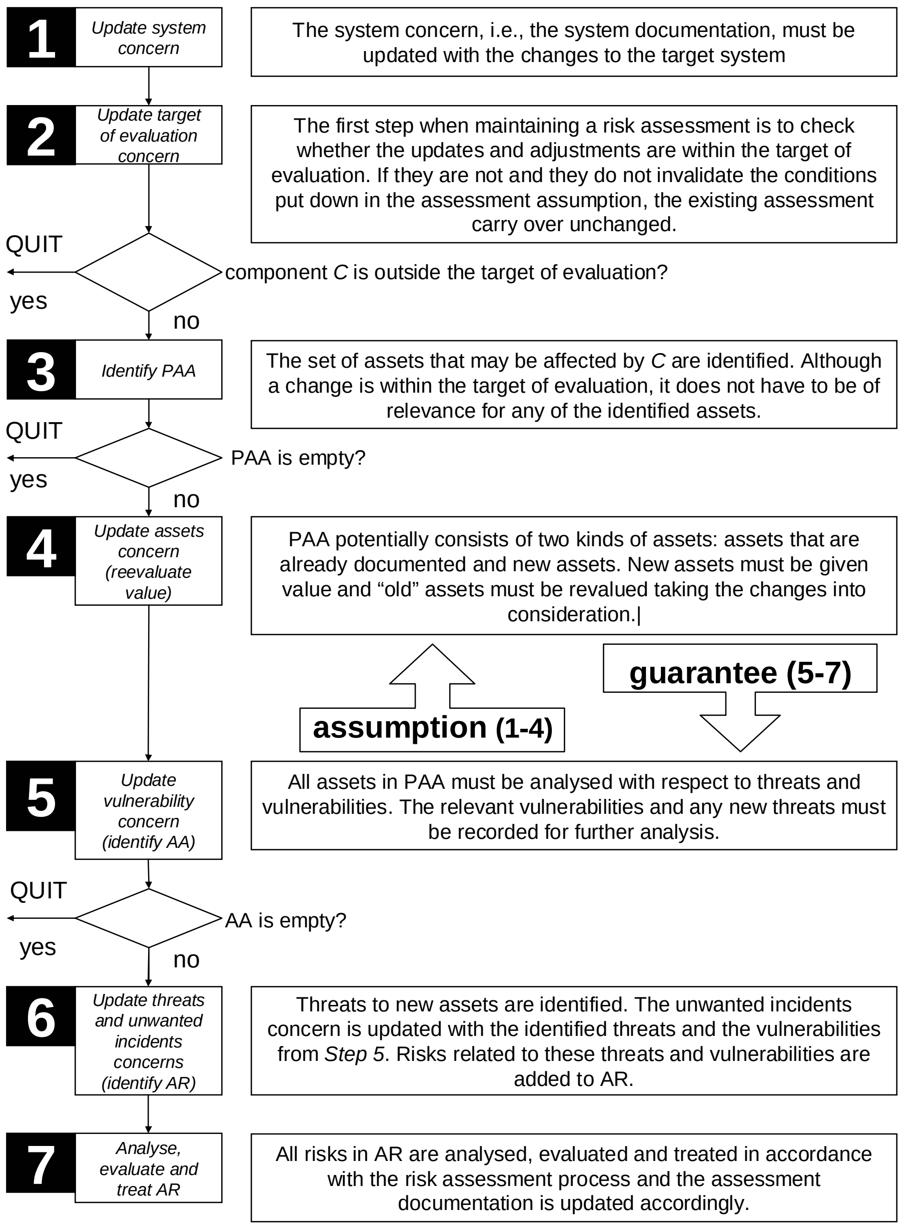
1 Update system concern
The system concern, i.e., the system documentation, must be updated with the changes to the target system
2 Update target of evaluation concern
The first step when maintaining a risk assessment is to check whether the updates and adjustments are within the target of evaluation. If they are not and they do not invalidate the conditions put down in the assessment assumption, the existing assessment carry over unchanged.
QUIT
yes
component C is outside the target of evaluation?
no
3 Identify PAA
The set of assets that may be affected by C are identified. Although a change is within the target of evaluation, it does not have to be of relevance for any of the identified assets.
QUIT
yes
PAA is empty?
no
4 Update assets concern (reevaluate value)
PAA potentially consists of two kinds of assets: assets that are already documented and new assets. New assets must be given value and "old" assets must be revalued taking the changes into consideration.|
guarantee (5-7)
assumption (1-4)
5 Update vulnerability concern (identify AA)
All assets in PAA must be analysed with respect to threats and vulnerabilities. The relevant vulnerabilities and any new threats must be recorded for further analysis.
QUIT
yes
AA is empty?
no
6 Update threats and unwanted incidents concerns (identify AR)
Threats to new assets are identified. The unwanted incidents concern is updated with the identified threats and the vulnerabilities from Step 5. Risks related to these threats and vulnerabilities are added to AR.
7 Analyse, evaluate and treat AR
All risks in AR are analysed, evaluated and treated in accordance with the risk assessment process and the assessment documentation is updated accordingly.

Figure 5 General procedure

In the contract-oriented approach, the Steps 1-4 update the assessment assumption, while Steps 5-7 update the assessment guarantee. If the procedure ends by one of the two first branches, then the assumption is not affected by the changes in the system and it may therefore be concluded that the changes will not affect the assumption guarantee.

4.1.1 Component modification

If component C is modified to become component C' (see Figure 6; RA:$A+B+C$ is the result of assessing $A+B+C$) we may assume that the component either is outside target or that the component is already assessed in an earlier assessment. We further assume that the component (or its behaviour and features) is not changed in a fundamental manner.

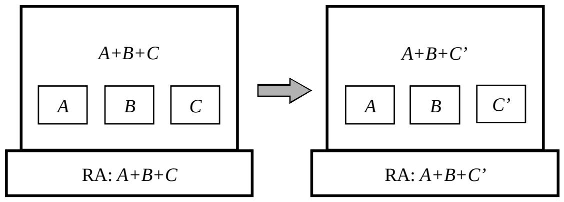

Figure 6: Maintaining component *C*

This means the general procedure for maintenance of assessment results outlined above may be simplified.

Step 1: Only small changes to the system concern are required. Descriptions of the internal states and the communication of the modified component may need updates, but, e.g., the infrastructure description of the system will remain unchanged.

Step 2: The target of evaluation may only be a certain part or feature of the component in question. Hence, the first step when maintaining assessment results for a component undergoing minor updates is to carefully inspect the target of evaluation concern and assess whether the updates are of relevance for the target of evaluation. If this assessment concludes they are not, the existing assessment results carry over unchanged and the procedure ends.

Step 3: Since no major changes to the target system are made, we may assume that there are no new assets and that there is no need for removing assets from the assets concern.

Step 4: According to the assumption made in *Step 3*, there are no new or removed assets in PAA. Hence, the updates in the assets concern only concern the value of the assets.

Step 5: The assets of PAA must be assessed with respect to threats and vulnerabilities. There are reasons to believe that no new external threats are present, but there may be new internal ones. The modification of a component may lead to new vulnerabilities, may remove vulnerabilities or, most likely, may result in changed frequency and consequence estimates.

Step 6: Most likely already identified threats may exploit new vulnerabilities of the assets in AA giving rise to new unwanted incidents, but new any new threats must also be considered. If new unwanted incidents are identified, the unwanted incidents concern must be updated.

4.1.2 Component substitution

Consider the situation (illustrated in Figure 7) where a system for which we have already carried out an assessment is updated by replacing one of its components by a new component. Assume we have a system *A*+*B*+*C* consisting of three components *A*, *B* and *C*. Moreover, assume we have carried out an assessment for *A*+*B*+*C* documented by the assessment RA:*A*+*B*+*C*. If component *C* is replaced by component *D* then we would like a strategy for making use of the assessment RA:*A*+*B*+*C* to arrive at the assessment RA:*A*+*B*+*D* of the new system *A*+*B*+*D*. We consider both the situation where *D* is not yet assessed and the situation where an assessment of *D* already exists.

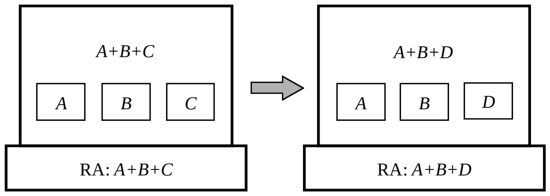

Figure 7: Replacing component *C* by *D*

We may assume that *D*, even though it may have features different from *C*, does not change the system in a fundamental manner. We support this assumption by the observation that since D replaces C, it must in many respects be similar to C.

Step 1: The system concern must be updated to reflect the replacement of the component. Only the description of the replaced component and the external communication of the new component should have to be changed, while the infrastructure description of the system should remain unchanged.
Step 2: See *Step 2* of Section 4.1.1.
Step 3: Most likely the set of assets remains unchanged. Features of *D* that are not features of *C* may however constitute new assets.
Step 4: Since there most likely are no new assets, the main activity of this step is to re-evaluate the values of the assets of PAA. This is required regardless of whether assessment results exist for *D*, since values are stakeholder specific and in the general case *D* is not assessed for all stakeholders of the system.
Step 5: Because of the assumed similarities between *C* and *D*, there is reason to believe that there will not be any new external threats. An exception (which may not be the only one) is the case where *D* (and *C*) is part of the system's interface towards the external environment and *D* introduce new features to the system (e.g., a user interface that get new functionality). In any case, there may be new internal threats. Since the component *D* is new, it is reasonable to assume that the vulnerabilities of the assets in PAA may have changed. If there are assessment results for *D*, vulnerabilities related to *C* may be replaced with the vulnerabilities from this assessment. Great care is however required, since assets may be relevant for more components than *C*. If no assessment results exist for *D*, the documentation of vulnerabilities related to *C* is important input to the assessment of *D*, since it is also reasonable to assume that there are similarities between *C* and *D*.

4.1.3 Component introduction

In this section we consider the situation where a new component is added to the system. The situation is illustrated in Figure 8.

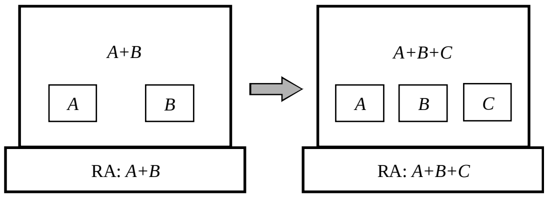

Figure 8: Adding component *C*

Step 1: The system concern must be updated to reflect the extension of the system. This means the system documentation must be extended with documentation of the new component and the interaction between the new component and the rest of the system. In this situation we cannot assume that there will be no changes in the infrastructure of the system.
Step 2: When a new component is added, we cannot schematically investigate whether the target of evaluation is changed. The case may be that the new component does not naturally fall either on the inside or the outside of the target. In this case the assessment team and the stakeholders have to decide whether the component is within target or not.
Step 3: Since the component is new, the component will normally provide the system with new features. We must therefore assume that there are new assets, even though the case may also be that *C* constitutes a part of an existing asset.
Step 4: Both new and "old" assets in PAA must be revalued. As in *Step 4* of Section 4.1.2 this is required independent of whether there are existing assessment results for *C*.
Step 5: Since *C* is a new component, addition of *C* may result in new threats, but it is reasonable to assume that no threats will disappear. The new component may however affect the consequence and frequency values of already identified risks. If assessment results exist for the new

component, most of the identified threats of these results will probably be relevant and should be added to the assessment documentation for the system. There may however be threats that are not relevant due to the placement of the component in the system. For this reason all threats of the pre-existing documentation of *C* should be assessed with respect to relevance.
When assessing the vulnerabilities of the affected assets, we distinguish between the new assets (the assets that are of relevance for *C* only) and the "old" assets (the assets that involve other components as well). For the first group of assets we may use assessment results for *C* directly if such results exist, or the assets may be assessed independently of the rest of the system. For the second group we have to be more careful, since the new component may have both positive and negative effects with respect to vulnerabilities. Consider for example the case where the asset is a network and the new component is a firewall. For the assets of the second group a new assessment of vulnerabilities is required, but both the original vulnerabilities and the vulnerabilities related to the new component are important inputs to this assessment.
Step 7: The new component may itself represent a treatment of documented risks to the system. This has to be taken into consideration when the concerns of Sub-processes 4 and 5 are updated.

4.2 Composing assessment results

Figure 9 illustrates the situation where two components, *A* and *B*, for which we have assessment results RA:*A* and RA:*B*, respectively, are composed.

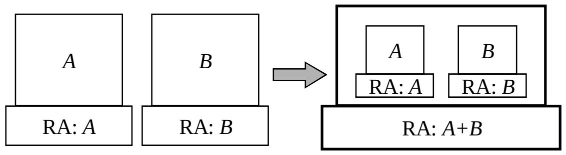

Figure 9: Composing assessment results

The general case of deducing an overall assessment RA:*A*+*B* from the existing assessments RA:*A* and RA:*B* is complex due to the possibility of circular argumentation. The assessment guarantee of RA:*B* may fulfil the assessment assumption of RA:*A* only if the assessment guarantee of RA:*A* fulfils the assessment assumption of RA:*B*, and the other way around. As explained for specification composition in [1], there are cases where such dependency cycles can be broken down and valuable information on the composed system be extracted. CORAS procedures for interesting special cases have been developed. Procedures that handle the general situation are an issue for ongoing research.

4.3 Reusing assessment results

Traditionally, system development methodologies focus on the development of single systems. More recently, the emphasis has shifted towards the development of system product lines. Inspired by [8], we define a system product line, as a set of systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core entities in a prescribed way. According to [3], components provide the perfect foundation for the practical application of product line development. Other examples of product-line oriented system development methods are FODA [25], FAST [36] and TIMe [8]. In the following we outline CORAS support for reuse of assessment results in a product-line oriented system development.
A critical factor in the success of a product line approach is the nature of the reusable "core". As argued in [3], "… at a minimum this should contain a reference architecture supported by techniques for capturing and selecting the points of variation among family members. In its most general form, the reusable asset takes the form of a framework, which embodies implementation

(i.e., code level) artefacts for the common parts of the family, as well as higher-level design and architecture models. Since they embody every possible reusable asset, at all levels of abstraction, frameworks constitute the largest possible reusable artefacts for a particular product family."
The CORAS framework is intended to support a product-line approach to system development in the following sense.

- For each component we may store assessment relevant information. When a developing a new product this assessment relevant information will be extracted for each component to be reused.
- If the particular use of a component requires maintenance, the assessment relevant information will be updated as outlined in Section 4.1.1.
- If the particular use of a component requires substitution, the assessment relevant information will be updated as outlined in Section 4.1.2.
- If the particular use of a component requires introduction, the assessment relevant information will be updated as outlined in Section 4.1.3.
- To the extent the new product involves composition; a security assessment of the composite component will be performed using procedures for composition currently under development (see Section 4.2).
- To the extent the new product also requires the development of completely new components from scratch, the reassessment of early assessment results at a later point in the development may benefit from the strategies of Sections 4.1 and 4.2. With respect to UP, this facilitates, for example, reuse of assessment results from the inception phase during the elaboration phase.

5. Conclusions

Structured documentation of assessment results and the assumptions on which they depend provides the foundation for maintenance as well as for a component-based approach to security assessment.
For details on the overall CORAS approach we refer to [6], [38]. The CORAS UML profile is further described in [16]. Experiences from the use of CORAS in major trials within e-commerce and telemedicine are documented in [33] and [35], respectively.
There are other approaches to model-based risk assessment; see for instance CRAMM [3], ATAM [10], SA [37] and RSDS [26]. The particular angle of the CORAS approach with its emphasis on security and risk assessment tightly integrated in a UML and RM-ODP is however new. In particular, the issue of maintenance and reuse of assessment results has received very little attention in the literature.
Contract-oriented specification has been suggested in many contexts and under different names. Within the RM-ODP community one speaks of contracts related to quality of service specification [14]. In the formal methods community there are numerous variations; the pre/post [15], the rely/guarantee [24] and the assumption/guarantee [1] styles are all instances of contract-oriented specification. Other more applied examples are the design-by-contract paradigm, introduced by Bertrand Meyer [28], and the UML based approach advocated by Mingins/Liu [29].
Since 1990, work has been going on to align and develop existing national and international schemes in one, mutually accepted framework for testing IT security functionality. The Common Criteria (CC) [21] represents the outcome of this work. The Common Criteria project harmonises the European "Information Technology Security Evaluation Criteria (ITSEC)" [18], the "Canadian Trusted Computer Product Evaluation Criteria (CTCPEC)" and the American "Trusted Computer System Evaluation Criteria (TCSEC) and Federal Criteria (FC)". The CC is generic and does not provide methodology for security assessment. CORAS, on the other hand, is devoted to methodology for security assessment. Both the CC and CORAS places emphasis on semiformal

and formal specification. However, contrary to the CC, CORAS addresses and develops concrete specification technology addressing security assessment. The CC and CORAS are orthogonal approaches. The CC provides a common set of requirements for the security functions of IT products and systems, as well as a common set of requirements for assurance measures applied to the IT functions of IT products and systems during a security evaluation. CORAS provides specific methodology for one particular kind of assurance measure, namely security risk assessment.

References

[1] Abadi, M., Lamport, L. Conjoining specification. ACM TOPLAS 17:507-533,1995.

[2] AS/NZS 4360:1999 Risk management.

[3] Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wüst, J., Zettel, J. Component-based product line engineering with UML. Addison-Wesley, 2002.

[4] Barber, B., Davey, J. The use of the CCTA risk analysis and management methodology CRAMM. Proc. MEDINFO92, North Holland, 1589–1593, 1992.

[5] Bouti, A., Ait Kadi, D. A state-of-the-art review of FMEA/FMECA. International Journal of reliability, quality and safety engineering 1:515-543, 1994.

[6] den Braber, F., Dimitrakos, T., Gran, B. A., Lund, M.S., Stølen, K., Aagedal, J.Ø. The CORAS methodology: model-based risk management using UML and UP. Chapter in book titled UML and the Unified Process. IRM Press, 2003.

[7] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E. Extensible markup language (XML) 1.0 (Second edition). World Wide Web Consortium recommendation REC-xml, October 2000.

[8] Bræk, R., Gorman, J., Haugen, Ø., Melby, G., Møller-Pedersen, B., Sanders, R. Quality by construction exemplified by TIMe - the integrated methodology. Telektronikk 95(1):73-82, 1999.

[9] Clements, P., Northrop, L. Software product lines: practices and patterns. Addision-Wesley, 2001.

[10] Clements, P., Kazman, R., Klein, M. Evaluating software architectures: methods and case studies. Addison-Wesley, 2002.

[11] Cockburn, A. Structuring use cases with goals. Journal of object-oriented programming, Sep/Oct: 35-40, Nov/Dec: 56-62, 1997.

[12] D'Souza, F., Wills, A. C. Objects, components, and frameworks with UML: the Catalysis approach. Addison-Wesley, 1998.

[13] Dimitrakos, T., Ritchie, B., Raptis, D., Aagedal, J.Ø., den Braber, F., Stølen, K., Houmb, S-H. Integrating model-based security risk management into eBusiness systems development: The CORAS approach. In Proc. I3E2002, the 2nd IFIP conference on eBusiness, e-Commerce, e-Government. Kluwer Acadmic Publishers, October 2002. (To appear)

[14] Fevrier, A., Najm, E., Stefani, J. B. Contracts for ODP. Proc. ARTS97, LNCS, 1997.

[15] Hoare, C. A. R. An axiomatic basis for computer programming. Communications of the ACM, 12:576-583, 1969.

[16] Houmb, S-H., den Braber, F., Lund, M.S., Stølen, K. Towards a UML profile for model-based risk assessment. In Proc. UML'2002 Satellite Workshop on Critical Systems Development with UML, pages 79-91, Munich University of Technology, 2002.

[17] IEC 1025: 1990 Fault tree analysis (FTA).

[18] Information technology security evaluation criteria (ITSEC), version 1.2, Office for Official Publications of the European Communities, June 1991.

[19] ISO/IEC 10746: 1995 Basic reference model for open distributed processing.

[20] ISO/IEC TR 13335-1:2001: Information technology – Guidelines for the management of IT Security – Part 1: Concepts and models for IT Security.

[21] ISO/IEC 15408:1999 Information technology – Security techniques – Evaluation criteria for IT security.

[22] ISO/IEC 17799: 2000 Information technology – Code of practise for information security management.

[23] Jacobson, I., Rumbaugh, J., Booch, G. The unified software development process. Addison-Wesley, 1999.

[24] Jones, C. B. Development methods for computer programs including a notion of interference. PhD-thesis, Oxford University, 1981.

[25] Kang, K. C., Cohen, S. G., Novak, W. E., Peterson, A. S. Feature-oriented domain analaysis (FODA) feasibility study. Technical report UMIAC-TR-21, SEI, 1990.

[26] Lano, K., Androutsopoulos, K., Clark, D. Structuring and design of reactive systems using RSDS and B. Proc. FASE 2000, LNCS 1783, 97-111, 2000.

[27] Littlewood, B. A reliability model for systems with Markov structure. Appl. Stat. 24:172-177, 1975.

[28] Meyer, B. Object-oriented software construction. Prentice Hall, 1997.

[29] Mingis, C., Liu, Y. From UML to design by contract. Journal of object-oriented programming, April issue: 6-9, 2001.

[30] Misra, J., Chandy, K. M. Proofs of networks of processes. IEEE transactions on software engineering, 7:417-426, 1981.

[31] OMG-UML. Unified Modeling Language Specification, version 1.4, 2001.

[32] OMG. UML for QoS & Fault Tolerance. OMG document number: realtime/03-08-06, 2003

[33] Raptis, D., Dimitrakos, T., Gran, B.A., Stølen, K. The CORAS approach for model-based risk analysis applied to the e-commerce domain. In Proc. Communication and Multimedia Security Conference, 2002.

[34] Redmill, F., Chudleigh, M., Catmur, J. Hazop and software hazop. Wiley, 1999.

[35] Stamatiou, Y.C., Henriksen, E., Lund, M.S, Mantzouranis, E., Psarros, M., Skipenes, E., Stathiakos, N., Stølen, K. Experiences from using model-based risk assessment to evaluate the security of a telemedicine application. To appear in Proc. Telemedicine in Care Delivery, 2002.

[36] Weiss, D. M. and Lai, C. T. R. Software product line engineering: a family based software engineering process. Addison-Wesley, 1999.

[37] Wyss, G. D., Craft, R. L., Funkhouser, D. R. The use of object-oriented analysis methods in surety analysis. SAND Report 99-1242. Sandia National Laboratories, 1999.

[38] Aagedal, J.Ø., den Braber, F., Dimitrakos, T., Gran, B.A., Raptis, D., Stølen, K. Model-based risk assessment to improve enterprise security. In Proc. Enterprise Distributed Object Communication (EDOC’2002), pages 51-62, IEEE Computer Society, 2002.