STF90 A04015 – Unrestricted

REPORT

The CORAS Tool-Supported Methodology for UML-Based Security Analysis

Fredrik Vraalsen, Folker den Braber, Ida Hogganvik, Mass Soldal Lund, Ketil Stølen

SINTEF ICT

Cooperative and Trusted Systems

February 2004

SINTEF REPORT

SINTEF

SINTEF ICT

Address: NO-7465 Trondheim
NORWAY
Location Trondheim:
S.P. Andersens v 15
Location Oslo:
Forskningsveien 1
Telephone: +47 73 59 30 00
Fax: +47 73 59 43 02

Enterprise No.: NO 948 007 029 MVA

TITLE

The CORAS Tool-Supported Methodology for UML-Based Security Analysis

AUTHOR(S)

Fredrik Vraalsen, Folker den Braber, Ida Hogganvik, Mass Soldal Lund, Ketil Stølen

CLIENT(S)

The Research Council of Norway

REPORT NO. STF90 A04015	CLASSIFICATION Unrestricted	CLIENTS REF.	
CLASS. THIS PAGE Unrestricted	ISBN 82-14-03364-0	PROJECT NO. 40332800	NO. OF PAGES/APPENDICES 16/0
ELECTRONIC FILE CODE CORAS framework report.doc		PROJECT MANAGER (NAME, SIGN.) Ketil Stølen	CHECKED BY (NAME, SIGN.) Jan Øyvind Aagedal
FILE CODE	DATE 2004-02-10	APPROVED BY (NAME, POSITION, SIGN.) Bjørn Skjellaug, Research director	

ABSTRACT

The CORAS project has developed a complete framework providing methodology, processes, languages and tools for UML-based security risk analysis in an overall setting of system development and risk management. Models expressed in a specialized UML language are advocated as means to support communication and interaction during structured brainstorming sessions. Results and documentation from the security analysis are stored in a computerized repository facilitating reuse, presentation and consistency checking. The framework has been assessed through seven field trials in the e-commerce and telemedicine domains. The framework provides a library of reusable experiences containing more than 100 reusable elements gathered through the field trials.

KEYWORDS	ENGLISH	NORWEGIAN
GROUP 1	ICT	IKT
GROUP 2	Risk analysis, modelling, methodology, tool support	Risikoanalyse, modellering, metodikk, verktøystøtte
SELECTED BY AUTHOR		

TABLE OF CONTENTS

1 Introduction

The EU-funded CORAS project (IST-2000-25031), successfully completed in September 2003, has developed a tool-supported methodology for UML-based security (risk) analysis – in the following referred to as the CORAS framework or just the framework.

The CORAS framework is structured into four main parts; a terminology, a library, a methodology, and a tool. The terminology defines important concepts and terms from the domains of security, risk analysis and systems documentation. It provides the foundation for a uniform description and use of the CORAS framework. The library is divided into entities, storage structures and classification systems. The methodology consists of risk analysis techniques, processes and languages. The tool provides computerized support for performing the risk analysis in accordance with the methodology.

The CORAS *terminology* integrates terminology for security and risk analysis with terminology for system documentation. The terminology applied is taken from the standards that the CORAS risk management process is based upon. In particular, the security specific terminology is taken from ISO/IEC 13335: Information Technology - Security Techniques - Guidelines for the Management of IT-Security [10], complemented by ITU-T X.800 Security architecture for open system interconnection for CCITT applications [23], IEEE610 Standard Glossary of Software Engineering Terminology [30] and IEC 61508: Functional Safety of Electrical/ Electronic/ Programmable Electronic Safety-Related (E/E/PE) Systems [8]. The risk analysis specific terminology is taken from AS/NZS 4360:1999 Risk Management [1], complemented by IEC 61508 and British Standard BS4778 [31]. The system documentation terminology originates from the reference model for open distributed processing (RM-ODP) [28].

There are two libraries in the CORAS framework, called *repositories*, which store experiences from previous analyses and results from ongoing analyses. The *library* part of the framework defines the type of entities that can be stored in the libraries, the actual structure they are stored in, and the type of classification system they are structured according to. Using an analogy to a book library; the entity is "Book", the storage structure is "Shelves", and a classification system could be "classified according to ISBN-number". The libraries in the CORAS framework contain UML-models and other documentation, which are stored in a special package-structure and classified according to its meta-data. In addition, consistency constraints are implemented to ensure that the various data elements remain mutually consistent in case of updates.

The CORAS assessment repository stores results from the actual security analyses. The experience repository, on the other hand, supports the security analysis process by providing general reusable experience packages. By facilitating reuse, it helps the user avoid starting from scratch for each new analysis. Reusable experience packages contain UML-models, checklists, procedures and more.

The CORAS UML-based security analysis methodology integrates aspects from partly complementary risk analysis techniques and state-of-the-art system modeling methodology. The *methodology* part of the framework consists of three different sections: risk analysis techniques, processes, and languages. The language section defines common languages to support the methodology. The process section includes instructions for how to 'execute' the methodology, i.e. how to perform a security analysis. The risk analysis section presents different types of risk analysis techniques used in the methodology, like HazOp [17], FMEA [3], and FTA [7].

The CORAS methodology includes three languages: a UML-based specification language targeting security analysis, an XML [22] mark-up for exchange of security analysis data, and a vulnerability assessment report format. The methodology builds on a complete risk management process with aspects from system development processes. In addition, initial work has been done on integrating risk management and system development into one complete process. The tool in the CORAS framework is a computerized tool that supports the methodology and implements the assessment repository and the experience repository. Figure 1 displays the overall structure of the CORAS framework.

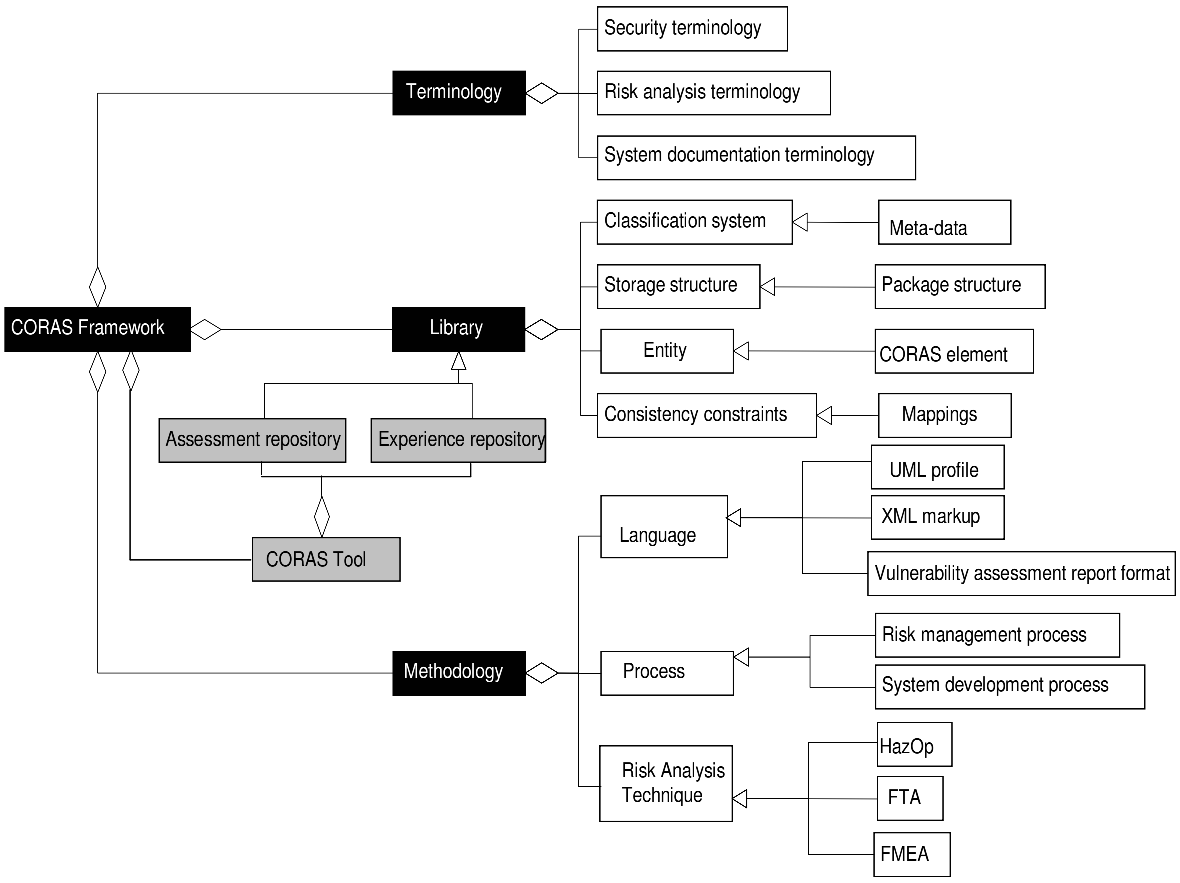

Figure 1: Overview of the CORAS Framework

This rest of the report is structured as follows: Section 2 introduces the UML profile for security analysis, while Section 3 describes the UML-based security analysis methodology supported by CORAS. In Section 4, we present the collection of reusable experience packages for security analysis from the CORAS field trials. Section 5 describes the computerized tool used to support the security analysis and explains its main design. Experiences from the field trials in CORAS are summarized in Section 6. Finally, Section 7 draws the main conclusions.

2 UML Profile for Security Analysis

The CORAS UML profile for security analysis [24][25] introduces a metamodel that defines an abstract language supporting model-based security analysis. Furthermore, the profile provides a mapping of classes in the metamodel to UML modelling elements by defining so-called stereotypes, and introduces special symbols (icons) for representing the stereotypes in UML diagrams. The motivation for the profile is to facilitate the practical use of UML to support security management in general, and security analysis in particular.

In the CORAS methodology, UML models are used for three different purposes:

- To describe the target of evaluation at the right level of abstraction.
- To facilitate communication and interaction between different groups of stakeholders involved in a security analysis.
- To document security analysis results and the assumptions on which these results depend in order to support reuse and maintenance.

The UML profile supports all these objectives, but has a special emphasis on communication and documentation. Documentation is supported since the metamodel of the profile is consistent with the data structure for security analysis documentation developed as part of the CORAS project. Communication is supported by the definition of easy-to-understand icons (symbols) associated with the modelling elements of the profile, and by these specialized modelling elements being consistent with the ontology of security analysis.

Methods for identification and analysis of security risks make use of structured brainstorming sessions. The effectiveness of such sessions depends on the extent to which the involved stakeholders and analysts are understood by each other. Since such sessions involve people with different backgrounds and competencies, such as users, system-developers, decision makers and system managers, common understanding among the stakeholders is often not the case. The typical use of the UML profile in such a setting is that the analyst leading the session presents diagrams for the rest of the participants. The participants need not be familiar with UML, to them the UML diagrams may be presented merely as illustrations of what they are discussing. For the analyst, however, the well-definedness of the UML models is of high importance, since it supports structured and uniform documentation of the security analysis results, as well as support for automated consistency checking and analysis.

The profile provides support for the risk management process by providing modelling support for:

- Security analysis context
- Assets
- Strengths, weaknesses, opportunities and threats (SWOT) analyses
- Threats and unwanted incidents
- Risk estimates
- Risk themes
- Treatments
- Treatment evaluations

Figure 2 illustrates the use of five stereotypes with icons from the security analysis profile; Treatment, UnwantedIncident, Asset, Attacker, and ThreatScenario. The asset in the diagram, in this case Availability of Service, represents one part of the system that needs to be protected. An identified threat to the service availability is flooding by a malicious person. The flooding may lead to a denial-of-service situation. The suggested treatment is a form of service authentication.

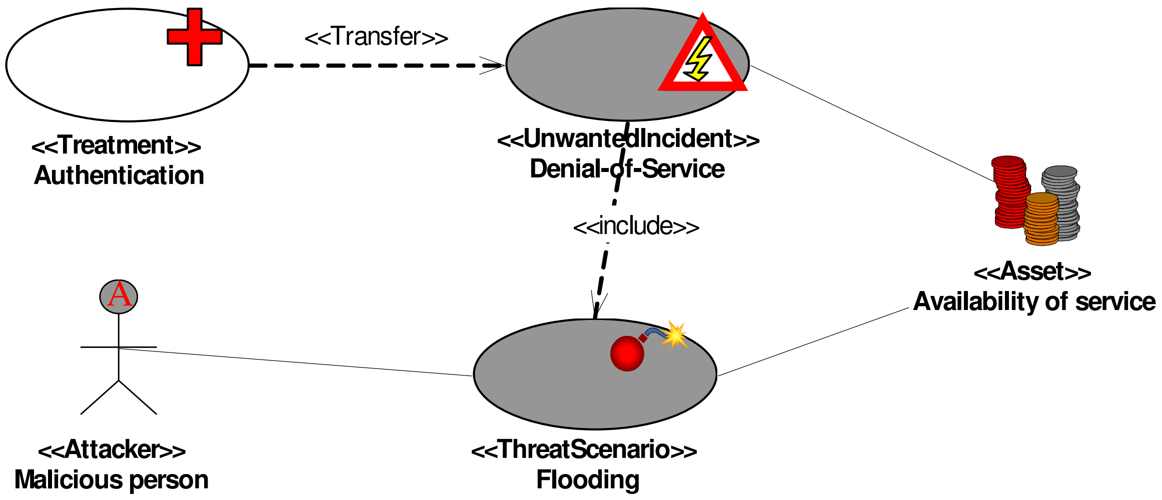

Figure 2: Example using the UML profile for Security Analysis

The profile has been developed parallel to the CORAS project and is influenced by continual feedback from a number of sources: from the development of the model-based security analysis methodology of CORAS, from use of the profile in large scale trials where the methodology was applied to real systems, as well as from the use of the profile in teaching of Master students. The profile is also a part of the proposal for "UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms" [27] that was adopted as a recommended OMG [14] standard in November 2003. The standardization process itself also provided much useful input to the development of the profile.

3 UML-Based Security Analysis Methodology

The CORAS methodology for UML-based security analysis applies UML [15] as a basis for security analysis activities within an overall process of risk management based on AS/NZS 4360:1999 "Risk Management" [1]. The CORAS methodology can be utilized on three abstraction levels; “decision maker”, “entry” and “full”. The “decision maker”-level is a high-level walk-through of the risk management process meant as an executive summary for decision makers. The “entry level” is a light-weight security analysis process intended for projects where there is limited competence or resources with regard to performing security analysis. The “full” version assumes that users have a high competence and experience within security analysis and risk management activities. For each level, process descriptions, recommendations and guidelines are provided, as well as templates, questionnaires and other supportive information.

The CORAS UML-based security analysis is about identifying negative functionality and making this *mis*functionality visible and known in order to protect the system. This misfunctionality can be everything from vulnerabilities that open for hacker-attacks to bad user interface design increasing the chance for crucial mistakes. The idea is that the well accepted techniques for specification and design of functionality are also suited to modelling of misfunctionality. This is illustrated by the misuse cases [19] in Figure 3.

Requirements analysis forms the basis for traditional modelling [4]. Security analysis has a similar role but focuses on undesirable behaviour. Every designed system contains risks and it is important that they are known. It is therefore not enough to only model the desired behaviour but also the undesired behaviour. In addition, the malicious actors, or “crooks”, need to be identified just like the normal actors. Model-based security analysis is about documenting the results of traditional risk analysis techniques in the same manner as we are used to for system requirements. The design process thus needs to take into account both desired and undesired behaviour and benevolent actors as well as malicious actors. As shown in Figure 3, modelling can be used for both aspects, providing a complete documentation of the system design including information about risks.

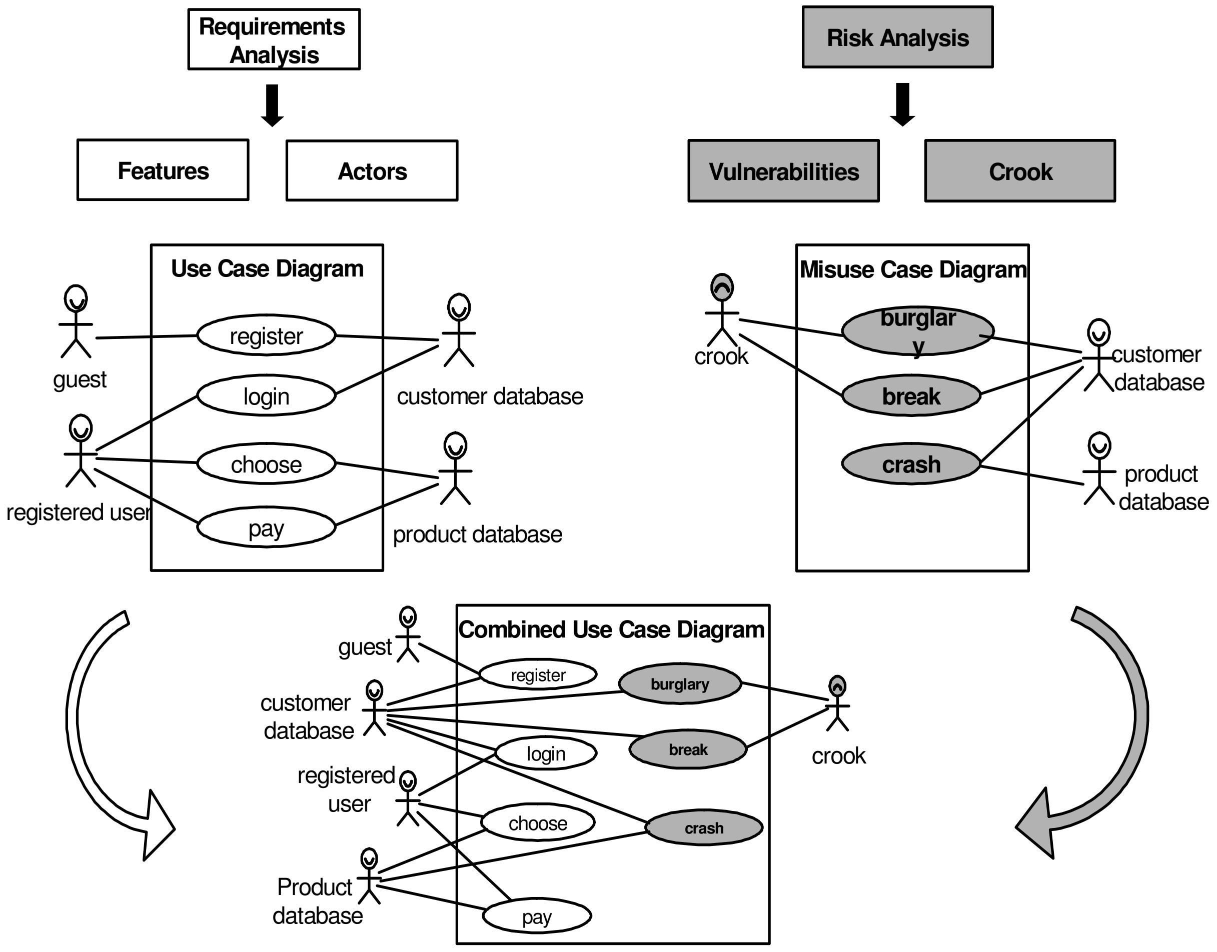

Figure 3: UML-based security analysis

4 Experience Repository

During field trials, security analysis documentation is produced in different forms. Most of the documentation will be specific to the system in question, but some results will be of generic interest, making them potentially reusable for later analyses.

The CORAS experience repository currently contains over a hundred reusable elements. These were extracted mainly from three sources: the security analysis documentation resulting from two major security analysis field trials carried out during the CORAS project, in addition to the experiences from the development of the CORAS methodology itself.

In the field trials, UML-diagrams were used to describe the target of evaluation, but results of different types like tables, check lists and natural language descriptions were also used. UML diagrams are either expressed in standard UML 2.0 or in the UML profile for security analysis described in Section 3.

Reusable experiences which logically belong together are grouped into experience packages. Experience packages serve multiple purposes in a model-based risk management process. They represent general patterns for security architectures or security policies. They also represent the generic parts of different classes of threat scenarios, as well as schemes for recording security analysis results and the assumptions on which they depend. Each experience may be applicable in several settings and can therefore be included in several experience packages.

The experience packages belong to a specific domain, e.g., e-commerce or telemedicine. Experience packages are classified as either constructive or supportive packages, which contain constructive and supportive elements, respectively, as illustrated in Figure 4. This classification is based on whether or not the element can be used directly as part of a security analysis result. Constructive elements become an actual part of the results from a security analysis, while supportive elements provide *support* in producing these results by capturing methodological aspects and patterns.

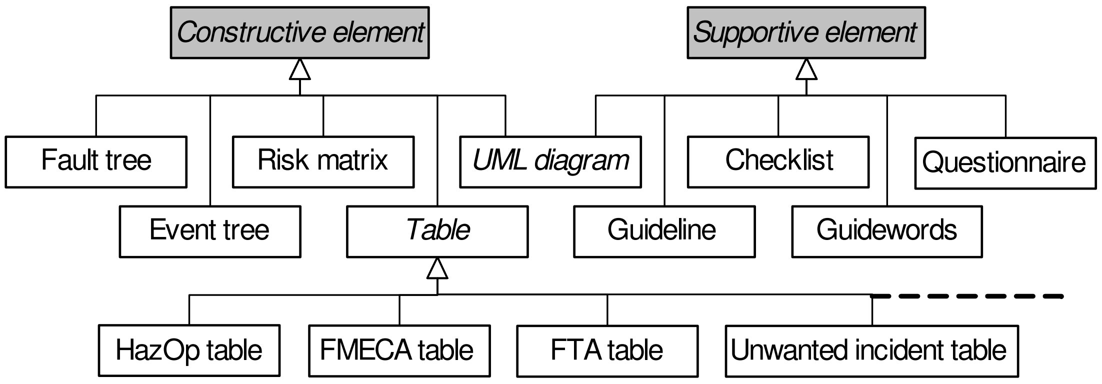

Figure 4: Elements

Constructive elements basically provide starting points when performing the security analysis. They provide templates to make the creation of security analysis results as easy as possible. Supportive elements on the other hand have a more guiding character. They might provide a recipe, or a helping hand, during either the instantiation of constructive reusable elements or the making of new reusable elements.

4.1 Structuring Experience Packages

The storage structure for security analysis results and reusable elements is referred to as the *CORAS documentation framework*. Each element in an experience package belongs to one *concern*, which relates to a specific security analysis activity. An example of a concern is the *assets concern*. This concern contains elements that may be used to support and assist asset identification. Asset identification is the activity of identifying anything that stakeholders find valuable and which must be protected. A concern may be defined as: *those interests, which relate to the system's development, its operation or any other aspect that are important to one or more stakeholders* [28].

Each element may also be related to one or more of the five *viewpoints* defined by RM-ODP [28], which means the element is especially interesting for a specific point of view of the system. RM-ODP defines a viewpoint as: *a form of abstraction achieved using a selected set of architectural concepts and structuring rules (a viewpoint specification), in order to focus on a particular concern within the system* [28]. The CORAS risk documentation framework divides the 5 RM-ODP viewpoint-structures into 22 concerns targeting security in general and model-based security analysis in particular. As indicated by Figure 5, concerns are cross-viewpoint perspectives linking together related information (elements) within different viewpoints.

Each element is classified according to its meta-data. Meta-data is information about the element like type, domain, concern, viewpoint, date of creation and what package it belongs to. Based on the meta-data it is possible to retrieve reusable elements in a consistent and structured manner.

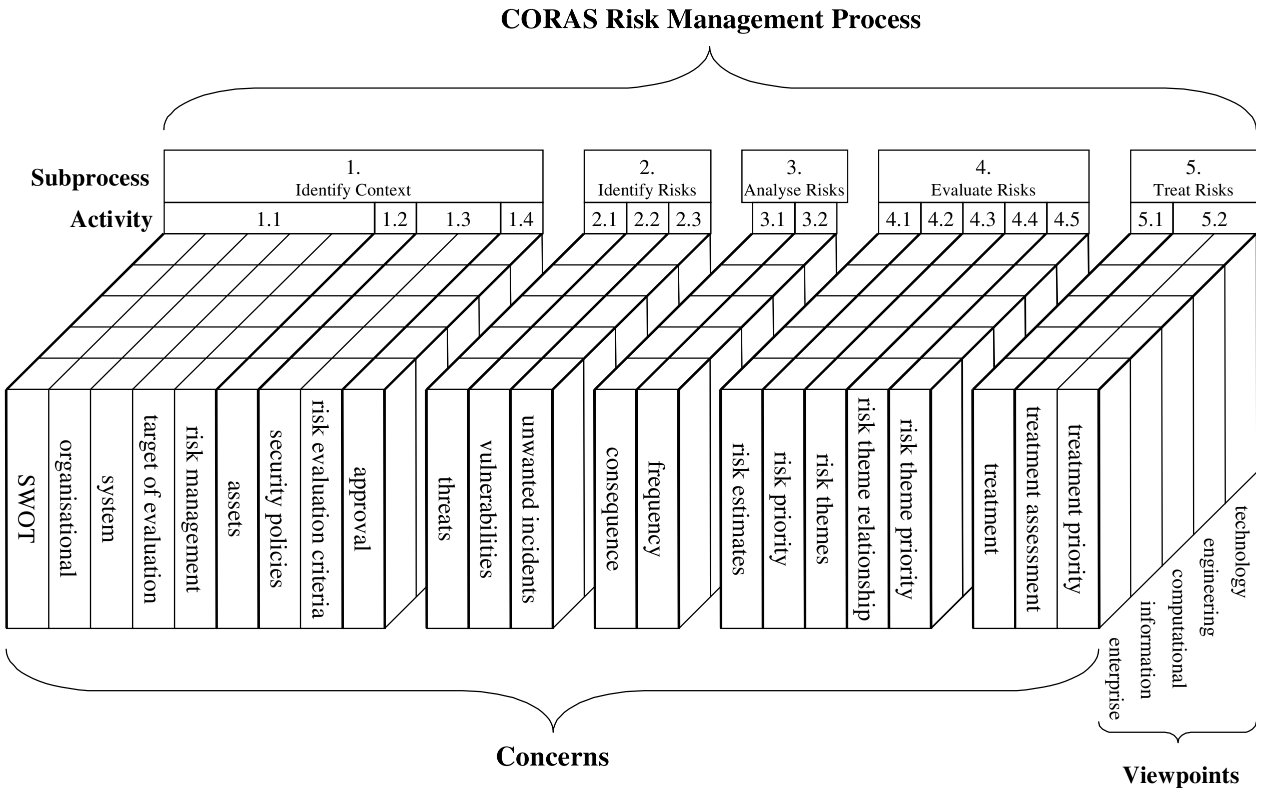

Figure 5: The Documentation Framework in CORAS

5 Tool Support

As should be clear by now, security analysis is a highly elaborate and prolific process which involves many different types of documentation. Future access to the information needs to be guaranteed, and the information needs to be shared between various modelling and security analysis tools. In addition, it is important that information stored in different places is consistent. Computerised support for managing the extraction, maintenance and reuse of security analysis results is of high importance.

A computerised tool based on the principles and requirements outlined above has been made publicly available as open source [5]. In the following we describe the overall structure and design of this tool.

Though the security management process is model-driven, tool support for security analysis data does not deal exclusively with models, but must also be able to include tables and other constructs such as event trees and fault trees. To satisfy this requirement, the tool is based on standardised, XML-based data formats, such as XMI for the interchange of UML models. The repositories are implemented on top of the open-source XML database eXist [26]. Figure 6 shows an overview of the CORAS tool. The tool is used to store the results from ongoing and completed security analyses, as well as the reusable experiences. As explained in previous sections, the *Assessment repository* is for analysis results, and the *Experience repository* for reusable experiences.

The tool GUI provides the end-user with administrative functionality, such as creating new security analysis projects and managing the reusable experience packages. Many entities of a security analysis, e.g. assets, stakeholders and threats, appear in several different security analysis results. It is therefore important to ensure that the various results are mutually consistent with regards to these entities.

A wide variety of UML modelling tools and risk analysis tools exist and are in use by security analysts and system engineers today. It is therefore important for our tool to provide flexible support for integration with external tools. To this end, the tool provides an integration layer with a defined API which other tools can use to integrate with the CORAS tool, utilising standardised XML formats for data integration. Figure 6 shows the components of this integration layer and the tool in more detail. Figure 6 defines three different viewpoints or roles. The tool *user* represents the end-user, e.g. a security analyst who uses the CORAS tool together with various modelling and risk analysis tools to support the security analysis process. The tool *integrator*, on the other hand, is responsible for integrating the various external tools with the CORAS tool, using the integration interface. Finally, the tool *developer* will implement the CORAS tool itself, and make sure that it provides the functionality specified by the integration interface.

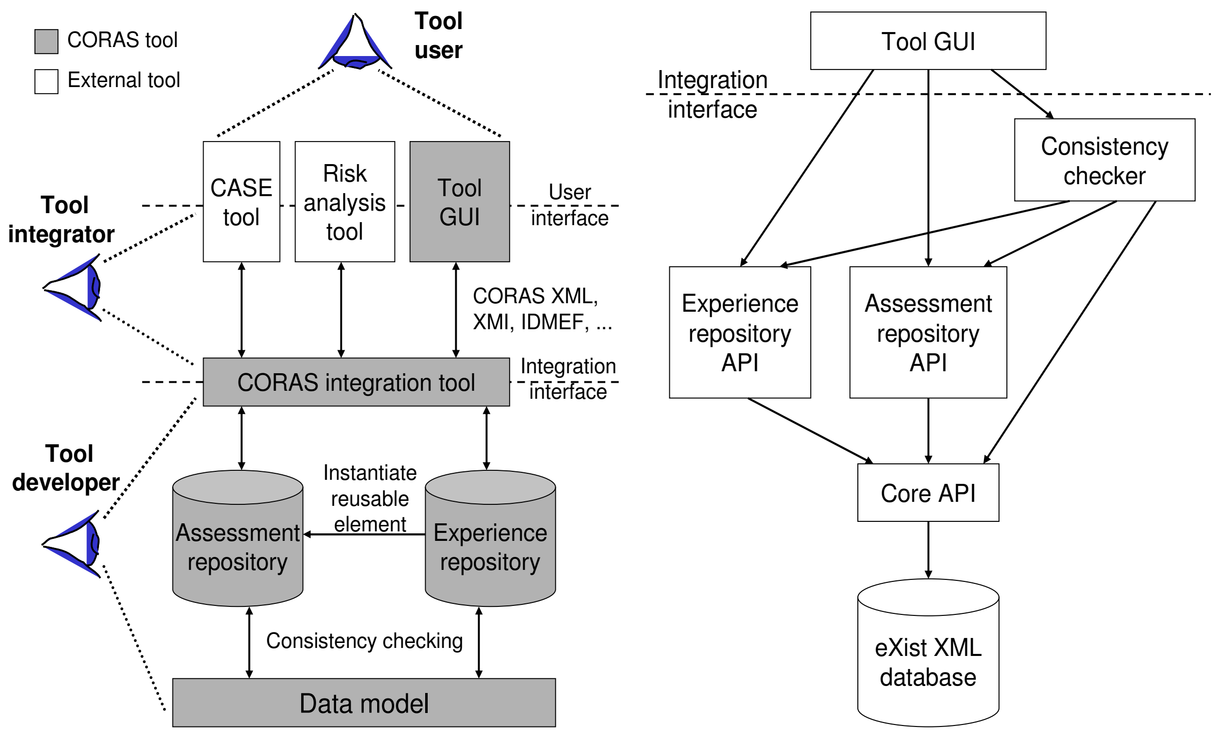

Figure 6: The Experience Repository

6 Field Trials

The CORAS project involved seven major field trials, each of which made use of the CORAS framework (as it was at the time of the field trial) to analyze the security of an already existing system or application. The field trials were all of industrial size. The largest involved an effort of more than 500 working hours. The field trials had four main objectives:

1. To ensure that the CORAS framework would be applicable to a wide range of security critical systems and applications.
2. To guide the development of the CORAS framework by providing feedback to the CORAS research activities throughout the lifetime of the CORAS project.

3. To benefit the trial sites whose systems and applications were targeted by the field trials.
4. To assess and evaluate the CORAS framework.

To meet Objective 1, the field trials were carried out within two different domains; three within e-commerce [6][16] and four within telemedicine [20][21]. A significant distinguishing factor between the e-commerce and the telemedicine systems and applications addressed in the trials was the nature of security risks they face. The target of the e-commerce field-trials was open to the Internet, attracting attackers that probe for weaknesses or opportunities for malicious exploitation. The telemedicine applications, on the other hand, operated on a closed network with authorized users communicating using computers in a controlled environment and with a controlled link to Internet (protected by firewalls).

To fulfil Objective 2, the research was carried out in three main iterations. Each main iteration was terminated by two field trials (three in the case of the third iteration); one focusing on telemedicine, and one targeting e-commerce. There was also a concluding phase after the field trials were completed. The iterative process provided the CORAS project an opportunity for testing the methodology as it was being developed and thus ensured that the methodology was practical in real life situations. Additionally, the field-trials provided the trial sites with significant information that enabled real life systems to be secured.

Each field trial was documented by two reports: a security analysis report and a report summarizing the experiences gained from using the tool-supported CORAS methodology within the field trial. The security analysis report presented the main findings and recommendations in a form suitable for the trial sites. Hence, this report was the main contribution with respect to Objective 3. The other report, the so-called experience report, was used to direct further research. The idea was that further research should address those aspects of the framework that the field trial in question identified as problematic from a practical point of view, and thereby help fulfil Objective 2. The experience reports also provided the foundation for the overall assessment of the CORAS framework required to meet Objective 4.

6.1 Assessment of the CORAS Framework

The CORAS framework was evaluated with respect to a set of evaluation criteria [29]. The basis for the evaluation was the experience reports from the seven field-trials. In the following we present some of the main conclusions from this evaluation.

It was concluded that the CORAS framework seems capable of addressing security concerns of web-based systems in general. The systems and applications targeted by the field trials were all analyzed successfully. Moreover, the CORAS framework was found to be very general since it gives the analyst much freedom to select analysis methods and modelling techniques depending on the target and the security issues to be analyzed.

Scalability was defined in this evaluation as the ability to analyze small, medium, as well as large and very complex systems [29]. It was found difficult to assess the scalability of the CORAS framework, given only the experiences from the seven field trials. However, there were no indications that the CORAS framework does not scale well. The e-commerce field trials focused on different but limited parts of the same e-commerce platform. In the case of telemedicine, the targets for the analyses were more complicated. The CORAS framework performed well in both situations.

In [29], it is argued that the CORAS framework is well-suited to identify the relevant kinds of risks due to its strong business and asset orientation. The CORAS methodology is asset-oriented in the sense that it starts by considering the enterprise level threats to assets through SWOT analysis and continues by exploring these enterprise level threats using FTA supported by HazOp and FMEA. The guidewords used during HazOp ensures that the different aspects of threats related to the target system are covered, while the tight integrations of assessment methods ensures that both high-level (enterprise level) and low-level threats (technical and information level) are addressed.

The capability of the CORAS framework to reduce costs through reusability of system documentation was also evaluated. In the second and third e-commerce field trial, although different parts of the system were under examination, many models were reused, especially models at a high level of detail. In the case of the telemedicine trials, the different applications that were analyzed were in the same network. This also facilitated much reuse of analysis documentation from one field trial to the next.

Special attention was given to the cost-efficiency of the CORAS framework. In [29], the costs associated with applying CORAS are contrasted with the benefits arising from applying CORAS. Based on data from the last telemedicine trial session, it was found that the benefits from applying the CORAS framework clearly outweighed the corresponding costs. This should be expected for other security-critical services as well. Estimations for the cost of using the CORAS framework was based on the effort and time needed for performing security analysis in one of the telemedicine field trials. The cost-benefit discussion hinges to some extent on whether or not the target of evaluation was already documented in UML.

7 Conclusions

We have presented the main technical results of the CORAS project. In particular, we have explained how these various technical results are combined in the overall CORAS framework. The CORAS UML profile for security analysis (that recently became a recommended OMG standard), the UML-based security analysis methodology, the experience repository and the tool support have received special attention.

The experiences from applying the CORAS framework in major field trials within e-commerce and telemedicine are promising. The applicability, usability and efficiency of the framework have been assessed through the field trials, and the overall assessment is documented in [29].

There are other approaches to model-based risk assessment; see for instance CRAMM [2], Surety Analysis [18], and OCTAVE [34]. We claim, however, that the CORAS framework has a number of features that to our knowledge is not matched in full by any other framework or methodology of this kind:

- The CORAS framework is fully compliant with UML 2.0. In fact, a specialized UML language for security analysis defined as a UML 2.0 profile has recently become a recommended OMG standard. There are a number of UML profiles targeting security, like UMLsec [32] and Secure UML [33]. Neither of these, however, provides specialized support for security risk analysis.
- The CORAS framework integrates a number of risk analysis techniques like HazOp, FMEA, FTA, etc. In the CORAS framework these techniques are integrated via an underlying data structure. The CORAS tool provides consistency checking across these various techniques.
- The CORAS tool comes with an XML API allowing the integration of tools for UML modelling, specialized risk analysis techniques, IDS, automatic vulnerability assessment, etc. The CORAS tool can be used to check consistency of or compare data generated via these various tools.

- The CORAS framework supports reuse of risk analysis documentation via a specialized experience repository. The repository is used to store generic models that can be instantiated, specialized or adapted into concrete descriptions required to support future analysis. The repository may also store methodological aspects in the form of patterns or procedures for how to conduct certain activities within a security analysis.
- A careful security analysis generates a lot of documentation. This documentation must be maintained when the assessed system is maintained. The CORAS framework supports this through its formal storage structure and facilities for consistency checking across different risk analysis techniques, and across risk analysis techniques and system documentation in the form of UML models.
- A security analysis identifies and documents the undesirable system behaviour. The CORAS UML profile employs the same kind of modelling techniques to document the undesirable behaviour that leading system development methodologies employ to capture requirements to the desirable system behaviour. For example, as pointed out above, in the same way as use cases are employed to capture desired functionality, misuse cases may be employed to capture threat scenarios. We believe that the fact that the CORAS methodology employs techniques well-known to system engineers simplifies security analysis during system development.

Acknowledgements

The research on which this report is based has partly been funded by the Research Council of Norway projects COBRA (152209/431), SECURIS (152839/220) and partly by the 5th Framework EU project CORAS (IST-2000-25031).The CORAS consortium consists of eleven partners from four countries: CTI (Greece), FORTH (Greece), IFE (Norway), Intracom (Greece), NCT (Norway), NR (Norway), QMUL (UK), RAL (UK), SINTEF (Norway), Solinet (Germany) and Telenor (Norway). The results reported herein have benefited from the joint efforts of the CORAS consortium.

References

[1] Australian/New Zealand Standard AS/NZS 4360 (1999). Risk management.
[2] Barber, B., & Davey, J. (1992). The use of the CCTA risk analysis and management methodology CRAMM. Proc. MEDINFO92, North Holland (pp. 1589 –1593).
[3] Bouti, A., & Ait Kadi, D. (1994). A state-of-the-art review of FMEA/FMECA. Interna-tional Journal of Reliability, Quality and Safety Engineering 1 (pp. 515-543).
[4] Cockburn, A. (1997). Structuring use cases with goals. Journal of Object-oriented Programming, Sep/Oct: 35-40, Nov/Dec: 56-62.
[5] CORAS (2000). A platform for risk analysis of security critical systems. IST-2000-25031. Accessed in 2003 from the World Wide Web: http://coras.sourceforge.net/
[6] Dimitrakos, T., Ritchie, B., Raptis, D., Aagedal, J.Ø., den Braber, F., Stølen, K., Houmb, S.H. (2002). Integrating model-based security risk management into eBusiness systems development - the CORAS approach. In Proc. 2nd IFIP Conference on E-Commerce, E-Business, E-Government (I3E'2003), Kluwer (pp. 159-175).
[7] IEC 1025. (1990). Fault tree analysis (FTA).
[8] IEC 61508. (2000). Functional safety of electrical/electronic/programmable safety related systems.
[9] ISO/IEC 10746.(1995). Basic reference model of open distributed processing.
[10] ISO/IEC TR 13335. (2001). Information technology – Guidelines for the management of IT security.
[11] ISO/IEC (1999). Information technology — Security techniques — Evaluation Criteria for IT Security ISO/IEC, 15408-1.
[12] ISO/IEC 17799.(2000). Information technology – Code of practice for information security management.
[13] Jacobson, I., Rumbaugh, J., & Booch, G. (1999). The Unified Software Development Process. Object Technology Series. Addison-Wesley.
[14] OMG. Accessed in 2003 from the World Wide Web: http://www.omg.org
[15] OMG. (2003a). Unified Modeling Language specification. Version 2.0. OMG document number: ptc/03-09-15.
[16] Raptis, D., Dimitrakos, T., Gran, B.A., Stølen, K. (2002). The CORAS approach for model-based risk analysis applied to the e-commerce domain. In Proc. Communication and Multimedia Security (CMS-2002), Kluwer (pp. 169-181).
[17] Redmill, F., Chudleigh, M., & Catmur, J. (1999). Hazop and software Hazop. Wiley.
[18] Sandia National Laboratories, Surety Analysis. Accessed in 2003 from the World Wide Web: http://www.sandia.gov
[19] Sindre, G., & Opdahl, A. L. (2000). Eliciting security requirements by misuse cases. In Proc. TOOLS_PACIFIC 2000. IEEE Computer Society Press (pp. 120-131).

[20] Stamatiou, Y., Skipenes, E., Henriksen, E., Stathiakis, N., Sikianakis, A., Charalambous, E., Antonakis, N., Stølen, K., den Braber, F., Soldal Lund, M., Papadaki, K., Valvis, G. (2003). The CORAS approach for model-based risk management applied to a telemedicine service. In Proc. Medical Informatics Europe (MIE'2003), IOS Press (pp. 206-211).
[21] Stathiakis, N., Chronaki, C., Skipenes, E., Henriksen, E., Charalambous, E., Sykianakis, A., Vrouchos, G., Antonakis, N., Tsiknakis, M., Orphanoudakis, S. (2003). Risk assessment of a cardiology eHealth service in HYGEIAnet. To appear in Proc. Computers in Cardiology (CIC'2003).
[22] World Wide Web Consortium. (6 October, 2000). Extensible Markup Language (XML) v1.0, W3C recommendation (second edition).
[23] ITU-T X.800, (1991): Security architecture for open system interconnection for CCITT applications. Geneva. (Technically aligned with ISO 7498-2)
[24] Lund, M. S., den Braber, F., Stølen, K., Vraalsen, F. A UML profile for the identification and analysis of security risks during structured brainstorming. Technical report STF40 A03067, SINTEF Telecom and Informatics, December 2003.
[25] Lund, M. S., Hogganvik, I., Seehusen, F., Stølen, K. UML profile for security assesssment. Techinical report STF A03066, SINTEF Telecom and Informatics, December 2003.
[26] Meier, W.: eXist: An Open Source Native XML Database. In: Akmal B. Chaudri, Mario Jeckle, Erhard Rahm, Rainer Unland (Eds.): Web, Web-Services, and Database Systems. NODe 2002 Web- and Database-Related Workshops, Erfurt, Germany, October 2002. Springer LNCS Series, 2593 (2002)
[27] OMG: UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms, Revised submission to OMG RFP ad/2002-01-07. OMG Document: realtime/2003-08-06. Object Management Group (2003)
[28] Putman, J. R., (2000): *Architechting with RM-ODP*, Prentice-Hall
[29] The CORAS Deliverable D5.15 *Trial results and assessment of the CORAS methodology.*
[30] IEEE610: IEEE Std 610.12 (1990): IEEE Standard Glossary of Software Engineering Terminology.
[31] British Standard BS4778 (1991): Quality vocabulary: Availability, reliability and maintainability terms. Glossary of international terms. British Standards Institute. http://www.bsi-global.com
[32] Jurjens, J. (2002). UMLsec: Extending UML for secure systems development. LNCS 2460.
[33] Lodderstedt, T., Basin, D., Doser, J. SecureUML: A UML-based modeling language for model-driven security. LNCS 2460.
[34] OCTAVE Information Security Risk Evaluation. Accessed in 2004 from the World Wide Web: http://www.cert.org/octave/