# A Component-oriented Approach to Security Risk Assessment

Mass Soldal Lund, Folker den Braber, Ketil Stølen

SINTEF Telecom and Informatics, Norway
{msl,fbr,kst}@sintef.no

**Abstract.** Security risk assessments are costly and time consuming and cannot be carried out from scratch each time a component is being reused in a new setting. This calls for a component-oriented approach tightly integrating security assessment in the system development and maintenance process. Such an approach requires a strategy for inferring useful information about the security of a composite component from the assessed security of its sub-components. The contribution of this paper is to provide such a strategy. The strategy makes use of specifications expressed in the form of contracts and builds on the Abadi/Lamport composition principle**.**

## 1 Introduction

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

- A careful integration of aspects from partly complementary risk assessment methods like HazOp[1] [31], FTA[2] [15], FMEA[3] [6], Markov analysis [21], and CRAMM[4] [5].
- Guidelines and methodology for the use of UML[5] [27] to support the security assessment methodology.
- A security risk management process based on AS/NZS 4360 [3] and ISO/IEC 17799 [18].
- A risk documentation framework based on RM-ODP[6] [16].
- An integrated security management and system development process based on UP[7] [19].
- A platform for tool-inclusion based on XML[8] [8].

[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.

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 [17]. 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.

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-definedness 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 by presenting a component-oriented approach to the reuse of assessment results. Components in our setting are not purely physical in the sense of contemporary "physical" component technologies (e.g., CORBA [37], .NET/COM+ [28, 29], J2EE/EJB [38]), but rather "logical" entities representing the logical building blocks of a system (as for example in Catalysis [11] and KobrA [4]). A component may itself consist of sub-components that may be viewed as independent components on their own.

To perform a security assessment of a single component is not different from assessing the security of a system given that the component is properly documented. Since the CORAS methodology is well documented in the literature, both from a theoretical perspective [7, 10, 14, 39] and in the form of field trials [30, 34, 35], we do not go into a description of this here. A component-oriented approach requires, however, a strategy for inferring useful information about the security of a composite component from the security assessments of its sub-components. The contribution of this paper is to provide such a strategy.

The remainder of this paper is structured into four sections. In order to integrate security assessment in a component-oriented approach to system development we need a basic understanding of how system development and security assessment

relate. Section 2 describes this relationship from three different perspectives: a modelling perspective, a procedural perspective and a contractual perspective. Based on this understanding, Section 3 outlines an approach to component-oriented security assessment. Section 4 provides a brief summary and draws the main conclusions.

## 2 System development versus security assessment

The processes of system development and security assessment are often carried out independently with little mutual interaction. Even today, security is typically first taken into consideration after the "real" system development has been completed. The result is expensive redesigns and unsatisfactory security solutions. As we argue from three perspectives in the following, these processes are in reality closely related and may be founded on the same kind of technologies.

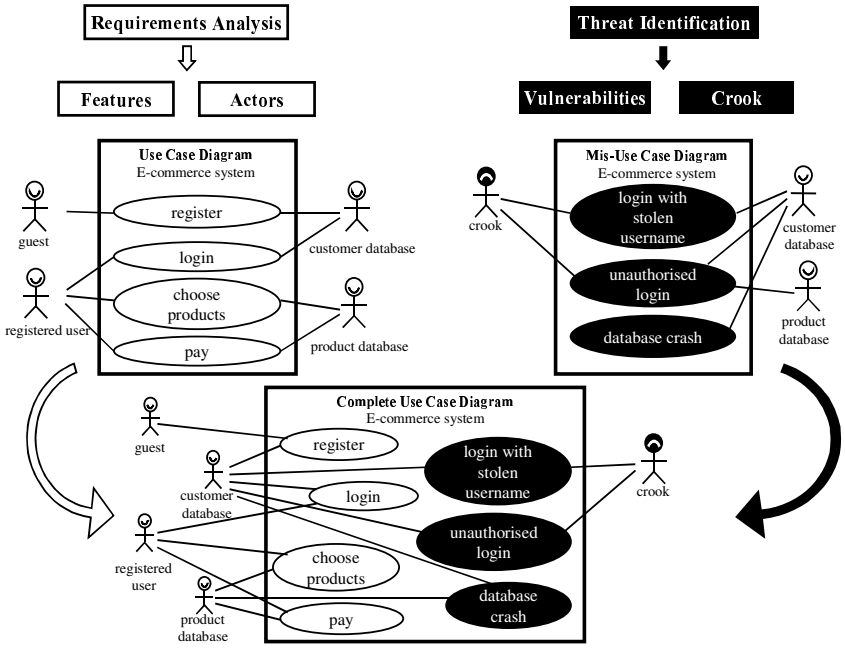


**Fig. 1.** Requirements capture versus threat identification

### 2.1 A modelling perspective

Requirements capture resembles risk identification in that both activities require different groups of stakeholders, e.g., users, decision makers, system developers, system managers etc., to communicate and reach consensus. Graphical specification

techniques like use-case and sequence diagrams have proved well suited to support this interaction in the case of requirements capture. CORAS employs the same kind of notations to facilitate threat identification. Fig. 1 highlights how threat identification mirrors requirements capture.

- In the case of requirement capture, the focus is on the *wanted* functionality. Use-case diagrams and sequence diagrams are often used to document this functionality.
- In the case of threat identification, the focus is on the *unwanted* functionality in the form of threat scenarios. For this purpose we may use misuse case diagrams [2, 33]. As in the case of use-cases, the behaviour of misuse-cases may be described using sequence diagrams.

The combination of these two views provides us with the "good" as well as the "bad" sides of the system under design. This "complete overview" is exactly what facilitates an integrated approach to system development and security.

### 2.2 A procedural perspective

In system development it is usual to distinguish between three levels of abstraction, the requirements specification, the design specification and the implementation. The design specification is a refinement of the requirement specification, and the implementation is a refinement of the design specification. Refinement means that any external behaviour allowed by the resulting more concrete system description is also allowed by the given more abstract system description.

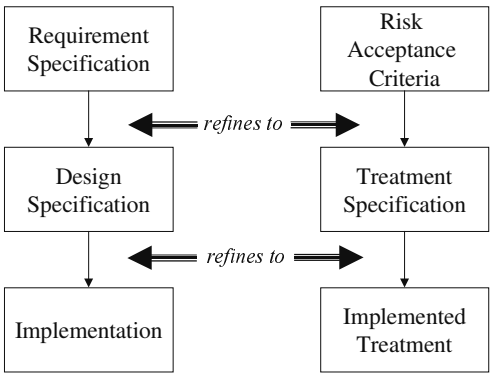


**Fig. 2.** System development versus risk treatment

As indicated by Fig. 2, these three levels of abstraction are mirrored by the security assessment process.

- The requirement specification corresponds to the risk acceptance criteria. They formalise what the customer may tolerate with respect to loss of assets.

– The design specification corresponds to the specification of the selected treatment describing how the system should be redesigned to fulfil the risk acceptance criteria.
– The implementation corresponds to the realisation of the treatment specification.

### 2.3 A contractual perspective

System components are not designed for arbitrary environments, but only for environments that satisfy certain assumptions. This motivates contract-oriented specification. 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 [12]. In the formal methods community there are numerous variations; the pre/post [13], the rely/guarantee [20] and the assumption/guarantee [26] specifications are all of contractual form. Other less formal approaches are the design-by-contract paradigm, introduced by Bertrand Meyer [24], and the UML based approach advocated by Christine Mingins and Yu Liu [25].

The contract-oriented specification is typically divided into two parts, an assumption and a guarantee:
– The assumption specifies the assumed environment for the specified component.
– The guarantee specifies how the specified component is guaranteed to behave whenever it is executed in an environment that satisfies the assumption.

As long as the environment behaves in accordance with the assumption, the component is required to fulfil the guarantee. If the environment breaks the assumption, the specified component may behave arbitrarily from then onwards.

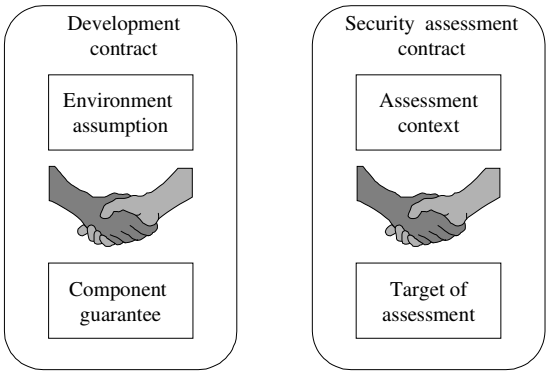


**Fig. 3.** Development contract versus security assessment contract

As indicated by Fig. 3, also a security assessment has a contractual flavour. A major objective of the context identification initiating any security assessment is to clearly specify the target to be assessed and the relevant properties of the surrounding environment as well as the assumptions on which the assessment is to be based. The

validity of the security assessment depends of course on the validity of the assessment context.

When performing a security assessment of a system, the target of assessment and assessment context will normally not correspond to the borderline between the system and the environment in which it is embedded. For example, if certain parts of the system are judged trustworthy (for example, the software for authentication or certain logging facilities), they may be viewed as part of the context for the assessment although they are tightly embedded in the software of the system to be assessed. In the case of a component to be reused in many systems it seems however reasonable to expect the target of assessment to be the full component.

## 3 Towards component-oriented security assessment

As argued above, to perform a security assessment of a single component is not much different from assessing the security of a system given that the component is properly documented. How the CORAS methodology can be used for this purpose is, moreover, well documented in the literature. In the following we therefore restrict our attention to the real challenge, namely to come up with a strategy for inferring useful information about the security of a composite component from the security assessments of its sub-components.

Fig. 4 illustrates the composition of two components, *A* and *B*, for which we have security assessment documentation AD:*A* and AD:*B*, respectively.

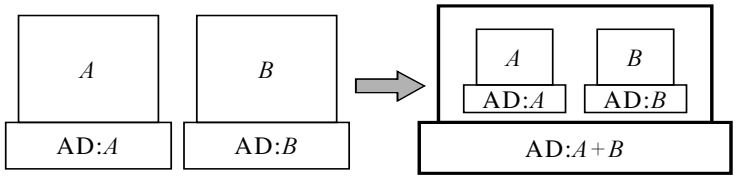


**Fig. 4.** Composing assessment results

To infer useful information AD:*A*+*B* about the security of the composite component from the security assessments AD:*A* and AD:*B* is nontrivial. Note for example that *B* "is in" the assessment context of *A*, and the other way around. Hence, the validity of AD:*B* may depend on that *A* fulfils security assumptions specified in the assessment context of *B*, and this may again depend on that *B* fulfils the specified assessment context of *A*. Hence, the argumentation may easily become circular. 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 can be extracted. Note also that although we for simplicity restrict ourselves to the binary case, the Abadi/Lamport principle is valid for any number of components and the same holds for our strategy.

### 3.1 The Abadi/Lamport principle

In [1] Abadi and Lamport formulate a rule for proving that the composite specification $(A_1,G_1)\|(A_2,G_2)$ of two interacting components is a refinement of the overall specification $(A,G)$, where $(A_1,G_1)$, $(A_2,G_2)$ and $(A,G)$ are contract specifications. The rule is formulated for the specification language Temporal Logic of Actions (TLA), but as argued in [9] this rule is an instance of a more general proof principle. Other instances of the rule are found in [20, 26, 36]. Although we in the following make use of a state-based formalism, the proposed approach may be reformulated in other settings, like in the stream-based approach of [36].

### 3.2 The basic semantic model

In order to make use of the Abadi/Lamport rule we adopt the semantic model of TLA. TLA is a state-based formalism, which means the behaviour of a component is characterised by sequences of states. The state of a component is an assignment of values to the external observable variables of the component. The occurrence of an event may change the state of the component, so from a given starting point we may build a tree of all possible states reachable by the component. This is called a behaviour tree, and each path in the tree we call a history. In Fig. 5 we see the top of a behavioural tree, where states are represented as nodes.

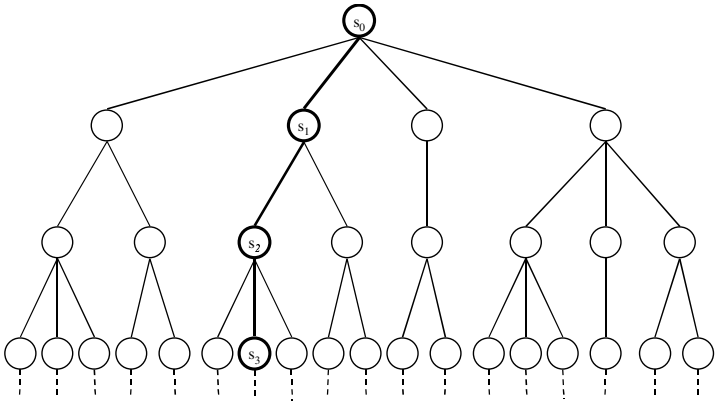


**Fig. 5.** Behaviour tree

The sequence $s_0, s_1, s_2, s_3, \ldots$ of states is one of several histories in the tree. Each $s_k$ is a tuple of values

$$s_k \in (U_1 \times U_2 \times \cdots \times U_n) \tag{1}$$

where $U_1$, $U_2$, …, $U_n$ are the types of the external observable variables of the component. Since the systems to which we will apply this formal model usually are non-terminating, we consider the tree to have infinite depth.

### 3.3 Extending the model to represent assets

For our purpose, the semantic model outlined above is not sufficient; security assessment carried out with the CORAS methodology is asset-oriented in the sense that any assessment is relative to identified assets. An asset is something a stakeholder regards as having value. This may be anything from an important piece of hardware to the reputation of a company. A component may be of relevance to a number of assets.

In the semantic model we represent assets as special kinds of variables. A state of the component is then an assignment of values to variables and assets, where the type of asset values is monetary. A state of a component with $n$ observable variables that is related to $m$ assets is then

$$s_k \in (U_1 \times U_2 \times \cdots \times U_n \times \$^m) \tag{2}$$

where \$ represent the set of all asset values.

In the following we let $s_k.a_i$ denote the value assigned to the $i^{\text{th}}$ asset in the $k^{\text{th}}$ state of the history $s$. If we in the history $s$ have that the $k^{\text{th}}$ state transition reduces the value of asset $a_i$, i.e., $s_k.a_i < s_{k-1}.a_i$ for some $k$, we say that there is a reduction in asset value, and that the prefix $s_0, s_1, \ldots, s_{k-1}, s_k$ of $s$ (in the following referred to as $s|_k$) is a threat scenario.

### 3.4 Extending the model with frequencies

We are now able to represent the consequence of a threat scenario as a reduction of asset value, but still there is something missing in the model. In the CORAS methodology risk values are represented by a combination of consequence and frequency.

It makes sense to assume that events happen with a certain probability, in other words that each time the behaviour tree branches there is a probability associated with each branch. If we know (or have good estimates of) these probabilities, we may use various methods for calculating the probability for reaching a particular state from the initial state (see, e.g., [32]). We let the probability of reaching a state $s_k$ via $s|_k$ become part of the state $s_k$, so we have

$$s_k \in (U_1 \times U_2 \times \cdots \times U_n \times \$^m \times P) \tag{3}$$

where

$$P = \langle 0 \ldots 1] \subseteq \mathbb{R} \tag{4}$$

is the set of all probabilities. We let $s_k.p$ denote the probability of $s|_k$. We obviously require that (1) a state transition will never increase the probability, and that (2) the probability of a node is equal to the sum of the probabilities of its direct descendant nodes. More formally:

$$\forall s \in S: \forall k \in \mathbb{N}: s_k.p \geq s_{k+1}.p \quad (5)$$

$$\forall k \in \mathbb{N}: \Sigma_{r \in R(s|_k)} r.p = s_k.p \quad (6)$$

where $S$ is the set of all histories in the behaviour tree, and $R(s|_k)$ is the set of direct descendants of $s_k$ with respect to $s|_k$.

### 3.5 Adapting the assumption/guarantee format

An assumption/guarantee specification is of the form

$$A \Rightarrow_+ G \quad (7)$$

where $A$ and $G$ are predicates over histories. The assumption $A$ expresses the assumptions made about the environment the component is supposed to live in, and the guarantee $G$ formalises the behaviour the component is required to fulfil whenever executed in an environment that satisfies the assumption. The "$\Rightarrow_+$" operator formalises that the guarantee is required to hold at least one computation step further than the assumption. The intuition is as follows:

- The component is required to obey the guarantee as long as the environment obeys the assumption.
- When the environment breaks the assumption then the component needs at least one step to react.

In the setting of security assessments we need to provide guarantees of another kind, namely guarantees with respect to the assessed security. In the CORAS methodology these guarantees are *acceptance criteria*. Acceptance criteria may for example be formulated along the lines of "There should be no risk with a loss greater than $X$ as consequence and a probability of happening greater than $Y$".

A security assessment also has a context: For example, the assessment context will specify the initial asset values, and the effect the context may have on asset values. Since assets are represented as (monetary) values in the semantic model, we may characterise assumptions on assets as predicates over histories.

This leads to a contractual specification of the form

$$A \wedge V \Rightarrow_+ G \wedge C \quad (8)$$

where $V$ and $C$ are predicates over histories expressing the assessment context and the acceptance criteria, respectively.

### 3.6 Lifting the composition principle

Since the extended assumption/guarantee format is of the same form as the usual format, and since assumption and the guarantee are still predicates over histories of states, the validity of the composition principle carries over straightforwardly.

### 3.7 Verification

As argued above the composition principle is valid for our extended semantics and assumption/guarantee specification format. We may therefore use the principle for verifying composition of security assessment contracts. There are various strategies for carrying out such verifications. The most obvious strategy is formal verification (verification by logical proof) since the composition principle is formulated as a logical inference rule. In, e.g., [1] this is the chosen strategy. In [22] it is explained how the composition principle can be applied for making efficient testing methods for verification of compositions. If the predicates are executable, we may use a similar strategy for the contractual specification format suggested above. Furthermore, if we represent the executable predicates by state-machines we may use model-checking techniques.

## 4 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. The contributions of this paper are as follows:

- We have introduced a general principle for inferring useful information about the security of a composite component from the security assessments of its sub-components.
- We have explained how this principle can be combined with various techniques for verification.

To achieve this we have adapted a basic interleaving model for concurrency to represent

- assets as a special kind of variables;
- consequences as reduction in asset value;
- threat scenarios as history prefixes;
- frequencies as the probabilities of threat scenarios.

We have adapted the assumption/guarantee format to represent the assessment context and the risk acceptance criteria. Furthermore, we have argued that the Abadi/Lamport composition principle remains valid, and that verification may be based on formal verification, model checking, testing, or a combination.

We are not aware of related approaches in the literature, with the exception of [23] that concentrates on maintenance and only touches on composition.

### 4.1 Acknowledgements

The research on which this paper reports has partly been funded by the Research Council of Norway IKT-2010 projects COBRA (152209/431) and SARDAS (15295/431) 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 in this paper have benefited from the joint efforts of the CORAS consortium.

## References

1. Abadi, M., Lamport, L. Conjoining specification. ACM TOPLAS 17:507-533 (1995)
2. Alexander, I. Misuse cases: Use cases with hostile intent. IEEE Software 20(1):58-66, (2003)
3. AS/NZS 4360:1999 Risk management.
4. 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)
5. Barber, B., Davey, J. The use of the CCTA risk analysis and management methodology CRAMM. In Proc. MEDINFO92, pages 1589–1593. North Holland, (1992)
6. 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)
7. 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 XVII in book titled UML and the Unified Process, pages 332-357, IRM Press (2003)
8. 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 (2000)
9. Cau, C., Collette, P. A unifying approach for shared variable and distributed message passing concurrency. Acta Informatica, 33:153-167 (1996)
10. 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, pages 159-175. Kluwer (2002)
11. D'Souca, F., Wills, A. C. Objects, components, and frameworks with UML: the Catalysis approach. Addison-Wesley (1998)
12. Fevrier, A., Najm, E., Stefani, J. B. Contracts for ODP. In Proc. ARTS97, LNCS 1231, pages 216-232. Springer (1997)
13. Hoare, C. A. R. An axiomatic basis for computer programming. Communications of the ACM, 12:576-583 (1969)
14. 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)
15. IEC 1025: 1990 Fault tree analysis (FTA).
16. ISO/IEC 10746: 1995 Basic reference model for open distributed processing.
17. ISO/IEC TR 13335-1:2001: Information technology – Guidelines for the management of IT Security – Part 1: Concepts and models for IT Security.
18. ISO/IEC 17799: 2000 Information technology – Code of practise for information security management.
19. Jacobson, I., Rumbaugh, J., Booch, G. The unified software development process. Addison-Wesley (1999)
20. Jones, C. B. Development methods for computer programs including a notion of interference. PhD-thesis, Oxford University (1981)
21. Littlewood, B. A reliability model for systems with Markov structure. Journal of the Royal Statistical Society Series C-Applied Statistics, 24(2):172-177 (1975)

22. Lund, M. S. Validation of contract decomposition by testing. Master's thesis, Department of Informatics, University of Oslo (2002)
23. Lund, M. S., den Braber, F., Stølen, K. Maintaining results from security assessments. In Proc. 7th European Conference on Software Maintenance and Reengineering (CSMR'2003), pages 341-350. IEEE Computer Society (2003)
24. Meyer, B. Object-oriented software construction. Prentice Hall (1997)
25. Mingis, C., Liu, Y. From UML to design by contract. Journal of Object-Oriented Programming, April issue: 6-9 (2001)
26. Misra, J., Chandy, K. M. Proofs of networks of processes. IEEE transactions on software engineering, 7:417-426 (1981)
27. OMG-UML. Unified Modeling Language Specification, version 1.4 (2001)
28. Platt, D. S. Introducing Microsoft .NET. Microsoft Press International (2001)
29. Platt, D. S. Understanding COM+. Microsoft Press International (1999)
30. 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. CMS-2002, pages 169-181. Kluwer (2002)
31. Redmill, F., Chudleigh, M., Catmur, J. Hazop and software hazop. Wiley (1999)
32. Ross, S. M. Introduction to probability models, 6th edition. Academic Press (1997)
33. Sindre, G., Opdahl, A. L. Eliciting security requirements by misuse cases. In Proc. TOOLS_PACIFIC 2000, pages 120-131. IEEE Computer Society (2000)
34. 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. In Proc. Telemedicine in Care Delivery, pages 115-119 (2002)
35. Stamatiou, Y. C., Skipenes, E., Henriksen, E., Stathiakis, N., Sikianakis, A., Charalambous, E., Antonakis, N., Stølen, K., den Braber, F., Lund, M. S., Papadaki, K., Valvis, G. The CORAS approach for model-based risk management applied to a telemedicine service. To appear in Proc. Medical Informatics Europe (MIE'2003)
36. Stølen, K. Assumption/commitment rules for dataflow networks — with an emphasis on completeness. In Proc. ESOP'96, LNCS 1058, pages 356-372. Springer (1996)
37. The common object request broker: architecture and specification. OMG (2001)
38. Wutka, M. Special edition using Java 2 enterprise edition (J2EE): With JSP, Servlets, EJB 2.0, JNDI, JMS, JDBC, CORBA, XML and RMI. Que (2001)
39. 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. EDOC'2002, pages 51-62. IEEE Computer Society (2002)