An Integrated Security Verification and Security Solution Design Trade-Off Analysis Approach

Siv Hilde Houmb
Department of Computer and Information Science,
Norwegian University of Science and Technology,
Sem Sælands vei 7-9,
NO-7491 Trondheim, Norway
Phone: +47 7359 3440
Fax: +47 7359 4466
sivhoumb@idi.ntnu.no

Geri Georg
Department of Computer Science,
Colorado State University,
601 S. Howes St., Fort Collins,
CO-80523-1873, USA
Phone: +1 970 491 6765
Fax: +1 970 49 2466
georg@CS.ColoState.EDU

Jan Jürjens
Systems Engineering,
TU Munich, Boltzmannstr. 3,
805748 München/Garching, Germany
Phone: +49 89 289 17338
Fax: +49 89 289 17307
juerjens@in.tum.de

Robert France
Department of Computer Science,
Colorado State University,
601 S. Howes St., Fort Collins,
CO-80523-1873, USA
Phone: +1 970 491 6356
Fax: +1 970 49 2466
france@CS.ColoState.EDU

An Integrated Security Verification and Security Solution Design Trade-Off Analysis Approach

Abstract

This chapter describes the integrated security verification and security solution design trade-off analysis (SVDT) approach. SVDT is useful when there is a diverse set of requirements imposed upon a security critical system, such as a required security level, time-to-market and budget constraints and end users' expectations. Balancing these needs requires developers to evaluate alternative security solutions, and SVDT makes this evaluation effective. UMLsec, an extension to UML for secure systems development, is used to specify security requirements, and UMLsec tools are used to verify if the alternative design solutions satisfy security requirements. Verified design alternatives are then evaluated by the security solution design trade-off analysis. The trade-off analysis is implemented using Bayesian Belief Nets (BBN) and makes use of a set of trade-off parameters, such as budget, business goals, and time-to-market constraints, to support design decisions regarding how to best meet security requirements while also fulfilling other diverse system requirements.

Keywords: Software Development, Decision Support Systems – DSS, Software Evaluation, IT Evaluation Methods, Bayesian Belief Networks (BBN), Security Verification, Aspect-Oriented Modeling (AOM), Software Architecture, Quality of Service, Security Management, Security Risk, Decision Models, Internet Security, Web-Based Applications; E-Commerce Models, E-Commerce Measurement

INTRODUCTION

Security critical systems must perform at the required security level, make effective use of available resources and meet end-user expectations. Balancing these needs, while fulfilling budget and time-to-market constraints, requires developers to design and evaluate alternative security solutions. Standards and techniques exist to aid in this work, but most address a single facet of a system, such as its development, or rely on specially trained assessors. For example, ISO 15408 Common Criteria for Information Technology Security Evaluation (ISO15408, 1999) is based on a qualitative assessment, performed by one or a few certified assessors, and focuses on system development activities. The Common Criteria certify security critical systems against seven Evaluation Assurance Levels (EAL) that define the security level of a target system. The Common Criteria do not capture the notion of a system's operational security level. This is the security level observed during its operation at a particular point in time, as described by Littlewood (1993). Trade-off techniques, such as ATAM (Kazman et al., 2000) and CBAM (Kasman et al., 2002) provide well-tested approaches to aid decision making at the architectural level. CBAM also takes economical implications into consideration during the analysis. However, these techniques rely on the presence of an assessor who is familiar with the technique and who has proper experience in multiple areas of architectural trade-off analysis. This makes it difficult for developers with little experience in the security domain, or with trade-off analysis processes, to consider alternative security solutions during system design. Evaluating solution alternatives during design is critical to providing security by design within budget constraints. Security and architectural evaluations are also, in many cases, time-consuming and resource intensive, which precludes their use in situations where time-to-market, budget and rapid incremental development constraints exist.

The integrated security verification and security solution design trade-off analysis (SVDT) approach described in this chapter takes into consideration both development and operational security level concerns by integrating security verification, risk assessment, design trade-off analysis and relevant parts of the Common Criteria. SVDT consists of four main steps: (1) identify potential misuses and assess their risks, (2) identify alternative security solutions for misuses that must be addressed, (3) perform UMLsec security verification of these security solutions to ensure they prevent the target misuse and (4) perform security solution design trade-off analysis among the alternative solutions.

This chapter focuses on steps (3) and (4), using UMLsec and its related tool-support to verify that potential security solutions do meet their intended requirements, and using a trade-off analysis tool that evaluates the relative ability of different solution designs to meet the complete set of system constraints. The trade-off tool incorporates the notion of a static security level to address the security level derived from development activities, and a dynamic security level to address the security level derived from predicting events when the system is operating. Determination of the static security level is based on the Common Criteria recommendations, while the dynamic security level is derived from information regarding both normal use and potential misuse of the system. The dynamic security level is estimated using a prediction model (Houmb et al., 2005b) that uses the following parameters: misuse frequency (MF), misuse impact (MI), security solution cost (SC) and security solution effect (SE).

The static security level, dynamic security level, standards, policies, laws and regulations, priorities, business goals, security risk acceptance criteria, and time-to-market (TTM) and budget constraints are all parameters to the trade-off analysis, which computes the Return on Security Investment (RoSI) for each security solution. To better facilitate the trade-off analysis and security verification, an aspect-oriented modelling (AOM) technique is used to separate security solutions from system core functionality. Security solutions are specified and analysed in isolation before examining their influence on the system. Inappropriate or insufficient solutions that fail security verification can be quickly discarded from continued consideration, while complete analysis is reserved for the most promising alternatives. The results of trade-off analysis of different design alternatives are compared using their Return on Security Investment (RoSI) value, and developers can make decisions regarding which alternatives best meet security requirements while also meeting other diverse system constraints.

The chapter is organised as follows. Background information on the techniques used in SVDT and related techniques is presented in the next section. This is followed by a discussion of the SVDT approach, concentrating on security verification and security solution design trade-off analysis. Both of these topics are presented in detail, in the context of a running example of the design of a portion of an e-Commerce prototype platform. Future trends in the design of security critical systems follow this discussion, and the chapter concludes with a discussion of the controlled development of secure systems.

BACKGROUND

Techniques used in SVDT to facilitate security verification and security solution design trade-off analysis are presented in this section. We also discuss other techniques often used in these areas and their relation to the SVDT approach.

The SVDT approach is based on model-driven development (MDD). MDD is a development method proposed by the Object Management Group (OMG) in which software development activities are carried out on models (OMG, 2003). The goal is to obtain the implementation of a system by transforming models from high levels of abstraction to lower levels of abstraction and eventually into code. Analysis and refinement can occur at any level of abstraction. Risk-driven development is used in conjunction with MDD to combine development and security risk management throughout system development. Model-based risk-driven development uses models to identify security risks, assess alternative treatments to these risks, and document the resulting design. The goal of model-based risk-driven development is to cost-effectively achieve a required level of security.

The two steps of the SVDT approach that are discussed in detail in this chapter are security verification of potential solutions and security solution design trade-off analysis among these potential designs. Both steps require that alternative design solutions be easily analysed and interchanged in the system design. For this reason the SVDT approach uses AOM techniques. AOM allows the separation of security solutions from system core functionality. Security solutions can then be analysed to verify their security properties without the added complexity of other

system components. The security solutions must be integrated (or composed) into the core system design for trade-off analysis, and AOM techniques also support this integration. Analysing an alternative solution is easily achieved by integrating it with the core system and performing the analysis again.

The AOM technique allows reuse of previous experience and domain knowledge since the security solution design aspects are completely reusable in their generic form. Aspects are UML template patterns specified using a specialised role-based modelling language and must be instantiated in the context of a system prior to composition (France et al., 2004a). The instantiation step includes binding the names of model elements in the aspect to those of comparable model elements in the core design. Model elements that are targets of composition are unrestricted as long as they are specified as targets in the aspect. Thus, classes, methods, attributes, relations, arguments, sequence fragments, messages, stereotypes and tags are all potential points where composition can occur. Composition consists of merging two models using default merging rules. Composition can result in adding model elements, deleting model elements, replacing model elements, or augmenting, deleting or replacing behaviour (France et al., 2004b). Composition directives can be used when it is necessary to change the default merging rules to obtain correct behaviour in the composed model (Straw et al., 2004). We have developed composition techniques for UML static structure diagrams and sequence behaviour diagrams. The composition of sequence diagrams is used in the SVDT approach. Our AOM techniques differ from others in two respects. First, many AOM techniques require that an aspect contain information regarding where and how in the system core functionality it will be applied (Jacobson, 2003a; Jacobson, 2003b; Kiczales et al., 2001), whereas our generic aspect models have no knowledge of any core system and are thus completely reusable. We create context-specific aspect models to specify system-specific information regarding where the aspect will be applied through model element name bindings. Secondly, other AOM techniques typically provide composition mechanisms to augment or replace model elements and behaviour (Clarke, 2002; Clarke et al., 2005), but we have also found the need to delete model elements and behaviour, as well as interleave behaviour. These capabilities are included in our AOM techniques.

Recall that in step (3) of the SVDT approach, security verification is performed on potential security solutions. SVDT uses security verification for two purposes. The first purpose of security verification is to verify that potential security design solutions fulfil their respective security requirements. The second purpose is to verify that these potential designs prevent the targeted misuse scenarios. Security requirements and potential design solutions must therefore be specified using a method that supports verification. Although there are many possible security verification approaches, we have chosen UMLsec because of its link to UML and the tool-support it provides (Jürjens, 2004). UMLsec eases the task of integrating AOM techniques, security verification and security solution design trade-off analysis through its capabilities. AOM techniques, UMLsec security verification and trade-off analysis are linked together in such a way that each builds on the other to make the process as efficient as possible. Security design solutions and security requirements are both specified using UMLsec, and this allows us to use UMLsec tool-support to verify whether potential designs meet their requirements. UMLsec is also used to model adversary behaviour in a misuse scenario, and again its tool-support can be used to verify whether a potential design solution prevents the adversary from successfully achieving the misuse. The security verification is thus used to ensure that only those security solutions that will fulfil the security requirements and sufficiently solve the misuse scenarios are evaluated in the trade-off analysis.

The security solution design trade-off analysis step of the SVDT approach utilises a specialised trade-off analysis tool that is based on Bayesian Belief Networks (BBN). Related trade-off analysis techniques exist, but none are tailored to the security domain. This does not mean that one cannot use these techniques when performing security solution design trade-off analyses, but rather that a specialised technique is more efficient, particularly for developers unfamiliar with the security domain. The main reason for this is that the effects of a series of security solution decisions are conceptually hard for humans to analyse due to the interrelations and dependencies between the security solutions. The two most often used software design trade-off analysis methods are the Architecture Trade-off Analysis Method (ATAM) and the Cost Benefit Analysis Method (CBAM).

In addition, there are variations on these two methods available. Both ATAM and CBAM were developed by the Software Engineering Institute (SEI). The focus of ATAM is to provide insight into how quality goals interact with and trade off against each other. ATAM consist of nine steps and aids in eliciting sets of quality requirements along multiple dimensions, analysing the effects of each requirement in isolation, and then understanding the interactions of these requirements. This uncovers architectural decisions, which are then linked to business goals and desired quality attributes.

CBAM is an extension of ATAM and looks at both the architectural and the economic implications of decisions. The focus in CBAM is on how an organisation should invest its resources to maximise gains and minimise risks. CBAM is still an architecture-centric method, but incorporates cost, benefit and schedule implications of decisions into the trade-off analysis.

The security solution design trade-off analysis presented in this chapter is a security specific design trade-off analysis. It incorporates ideas from both ATAM and CBAM and is extended to use security solution and misuse-specific parameters, in addition to the economic implications, as input to the trade-off analysis. It is also extended to evaluate the required security level by combining both static and dynamic security levels. Analysing the consequence of security solution design decisions is also supported, because a composed model containing both the system and security solution is utilised. Because the security solution design trade-off analysis is coupled with AOM techniques, analysing several different alternatives is easy to do and developers can test a wide range of possible alternatives in order to find the design that presents the best fulfilment of competing requirements.

THE APPROACH

System security consists of defining, achieving and maintaining the following properties: confidentiality, integrity, availability, non-repudiation, accountability, authenticity and reliability, as defined by the security standard ISO 13335: Information technology - Guidelines for management of IT Security (ISO13335, 2001). The integrated SVDT approach targets all security properties, however the example given in this chapter only considers the security property confidentiality, which is also referred to as secrecy.

SVDT consists of the following four steps: (1) identify potential misuses and assess their risks, (2) identify alternative security solutions for misuses that must be addressed, (3) perform UMLsec security verification of these security solutions to ensure they prevent the target misuse and (4) perform security solution design trade-off analysis among the alternative solutions. This chapter focuses on step (3), the security verification of security solutions, and step (4), security solution design trade-off analysis. Step (1), identify potential misuses and assess their risk, and step (2), identify alternative security solutions, are performed using the CORAS risk assessment platform (CORAS Platform, 2005). This platform is based on the CORAS model-based risk assessment methodology (Stølen et al., 2002). For the reminder of the chapter we therefore assume that potential misuses, their associated risk and potential alternative security solutions are already identified and assessed. Figure 1 shows an overview of the SVDT approach for steps (3) and (4). The alternative security solutions for a particular misuse are represented by the set A, where a represents one of the security solutions in A. Security solutions are modelled as security aspects using UMLsec notation and security verification is performed using UMLsec tools. Design trade-off analysis is only performed on a security solution if it passes the verification by the UMLsec tools.

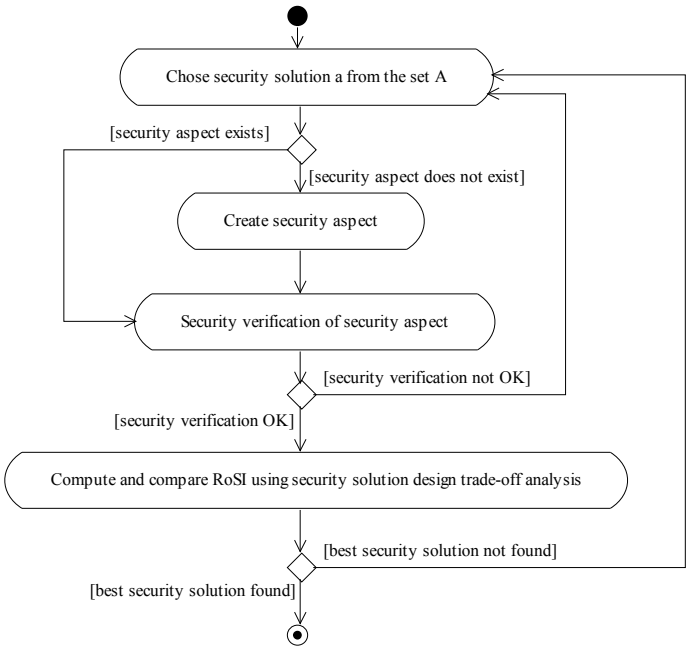

Figure 1. Overview of the security verification and security solution design trade-off analysis (SVDT) approach

Security Verification

The security verification performed in step (3) of the SDVT approach uses the UMLsec profile and tools (Jürjens, 2005). Potential security design solutions must be modelled as security aspects using the profile.

UMLsec

The UMLsec profile allows a developer to specify security-related information within the diagrams of a UML system specification. The profile uses the standard UML extension mechanisms *stereotypes*, *tagged values* and *constraints*. *Stereotypes* are used together with *tags* to formulate security requirements and assumptions related to the system environment, and *constraints* give criteria used to determine whether the requirements are met by the system design. Stereotypes define new types of modelling elements and extend the semantics of existing types or classes in the UML metamodel. Stereotype names are written in double angle brackets and are attached to a model element. The model element is then interpreted according to the extended meaning ascribed to the stereotype.
Properties are explicitly defined by attaching a *tagged value* to a model element. A tagged value is a name-value pair, where the name is referred to as the *tag*. The notation is *{tag=value}* with *value* to be assigned to the tag. Information can also be added to a model element by specifying *constraints* to refine its semantics. Stereotypes are used to attach tagged values and constraints as pseudo-attributes of the stereotyped model elements. Table 1 presents a fragment of the UMLsec stereotypes, together with their tags and constraints. More details can be found in Jürjens (2004).

Table 1. A subset of the UMLsec profile

Stereotype	Base Class	Tags	Constraints	Description
critical	object, subsystem	secrecy, integrity, authenticity		critical object or subsystem
data security	subsystem	adversary	secrecy, integrity and authenticity	basic data security requirements
secure links	subsystem	adversary	secrecy, integrity and authenticity matched by links	enforces secure communication links
LAN	link, node			LAN connection
Internet	link			Internet connection

Each stereotype in Table 1 can be applied to particular types of model elements. For example, the *critical* stereotype can be applied to an object or a subsystem, whereas the *LAN* stereotype must be applied to a link or a node. Specific tags can be associated with the stereotypes, e.g., *secrecy* can be associated with a *critical* subsystem, or *adversary* can be associated with a *secure links* subsystem. The *adversary* tag specifies a type of attacker (e.g., *default* or *insider*) and thus what type of adversary model should be used when verifying that a design meets its UMLsec specification. An adversary model specifies what capabilities an attacker has; for example, a default adversary has the ability to read, delete, insert and access information passed across an unencrypted Internet link.

Security Requirements for ACTIVE Modelled using UMLsec

The example used in this chapter is an e-Commerce platform prototype that provides electronic purchase of goods over the internet. The ACTIVE e-Commerce platform prototype was developed by the consortium of the EU project EP-27046-ACTIVE (ACTIVE, 2001). The design models presented here are a combination of refined models based on design descriptions provided by the ACTIVE project and new design models that take into consideration the result of the three risk assessments performed by the EU-project IST-2000-25031 CORAS (CORAS Project, 2005). For details on ACTIVE, the reader is referred to the publications of the two EU projects.

The infrastructure of the e-Commerce platform consists of a web server running Microsoft Internet Information Server (IIS), a Java application server (Allaire JSP Engine) and a database server RDBMS running Microsoft SQL server. The communication between the application server and the database is handled using the JDBC protocol. Potential customers access the web server from a Java enabled web browser using the HTTP protocol. The system deployment architecture is shown in Figure 2.

UMLsec, as described in the previous section, is used to model the security requirements. The security requirements can either be developed directly or derived from the results of a risk assessment giving misuses that need to be treated. In Figure 2 the communication link l_1 between the User PC and the e-Commerce Site is over the *Internet* using the communication protocol TCP/IP. The communication link l_2 between server-side components (e.g., web server and application server) occurs over a *LAN*. The UMLsec defined stereotypes <<*secure links*>>, <<*Internet*>> and <<*LAN*>> are used to indicate the type of communication link between nodes, and the stereotype <<*data security*>> is, as defined in the previous section, used to specify the requirements for data transmitted over the communication links. The meaning ascribed to the stereotype <<*data security*>> is specified in the class diagram in Figure 3.

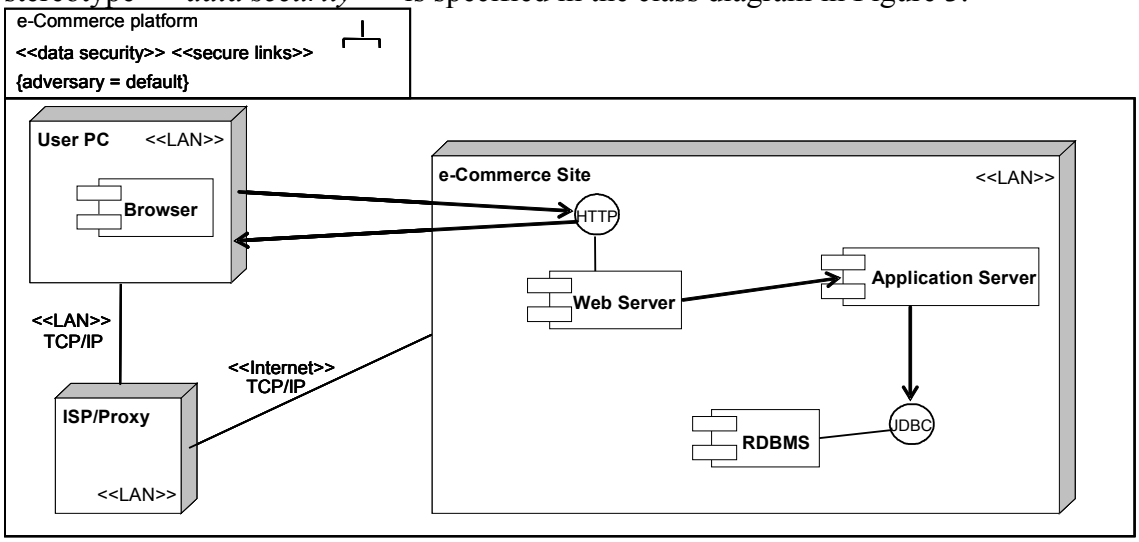

Figure 2. e-Commerce platform deployment diagram (refined from ACTIVE designs)

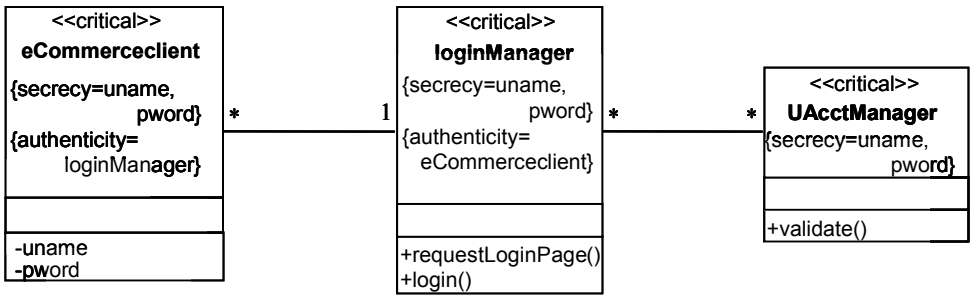

Figure 3. e-Commerce platform login service class diagram

The example in this chapter uses the login mechanism of the ACTIVE platform. To access any of the services in the e-Commerce platform a user must either login as a visitor or as a registered user. The UMLsec stereotype <<*data security*>>, specified for the subsystem *S* in the deployment diagram in Figure 2, is refined using the associated *tagged values* in the class diagram shown in Figure 3. The security requirement *"login information must be kept secret"* means that the user name, modelled as *uname* in the class diagram, and password, modelled as *pword* in the class diagram, must be kept secret. In order to preserve <<*data security*>> the communicating parties also need to know the authenticity of the other party, which is modelled using the associated tagged values *{authenticity=loginManager}* and *{authenticity = eCommerceclient}*.

After specifying the system using UMLsec, the list of misuses in need of treatment is modelled as adversary models. Adversary models describe potential adversary behaviour and thus the set of concrete threats against the system. Adversary models are functions of the form $\text{Threats}_M(s)$ that take an adversary type *M* and a stereotype *s* as inputs and return a subset of *{delete, read, insert, access}* as adversary capabilities. These capabilities are defined as follows: *delete* means that the adversary is able to delete data specified as protected by the UMLsec stereotypes, *read* means that the adversary is able to read these data, *insert* means that the adversary is able to insert additional data, and *access* means that the adversary is able to access protected data. These functions are derived from the specification of the physical layer of the system, which in this example is the deployment diagram in Figure 2. The risk assessment of the ACTIVE user authentication mechanism (Dimitrakos et al., 2002) identified the ACTIVE login process as being vulnerable to man-in-the-middle attacks. During this kind of attack user names and passwords can be intercepted by an attacker and used later to impersonate a valid user. This means that a default adversary (i.e., an attacker able to perform standard attacks) *M*, with communication stereotype *s*, has the set $\text{Threats}_M(s)=\{\text{read}\}$ for *s=Internet* and $\text{Threat}_M(s)=$ for *s=LAN*. Thus for communication link l_1*=Internet* in Figure 2, the result is that the default adversary *M* can *read* the secret data, *uname* and *pword*, that are specified in the class diagram of Figure 3.

One potential security solution for this misuse or adversary behaviour is to use transport layer security (TLS). Since the original TLS sequence includes potential vulnerabilities (Jürjens, 2004), a variant of TLS described by Jürjens (2004) is used. This security solution is then modelled as a security aspect using UMLsec, and the UMLsec tools are used to verify that it meets the security requirements and prevents the attack. This security verification occurs prior to evaluating the TLS security solution in the security solution design trade-off analysis.

TLS Aspect Security Verification

Security verification of the TLS aspect means checking whether the default adversary *A* will be able to obtain secret information. Figure 4 depicts the interaction template for the security aspect variant of TLS. More information on the TLS aspect can be found in Houmb et al. (2005c).

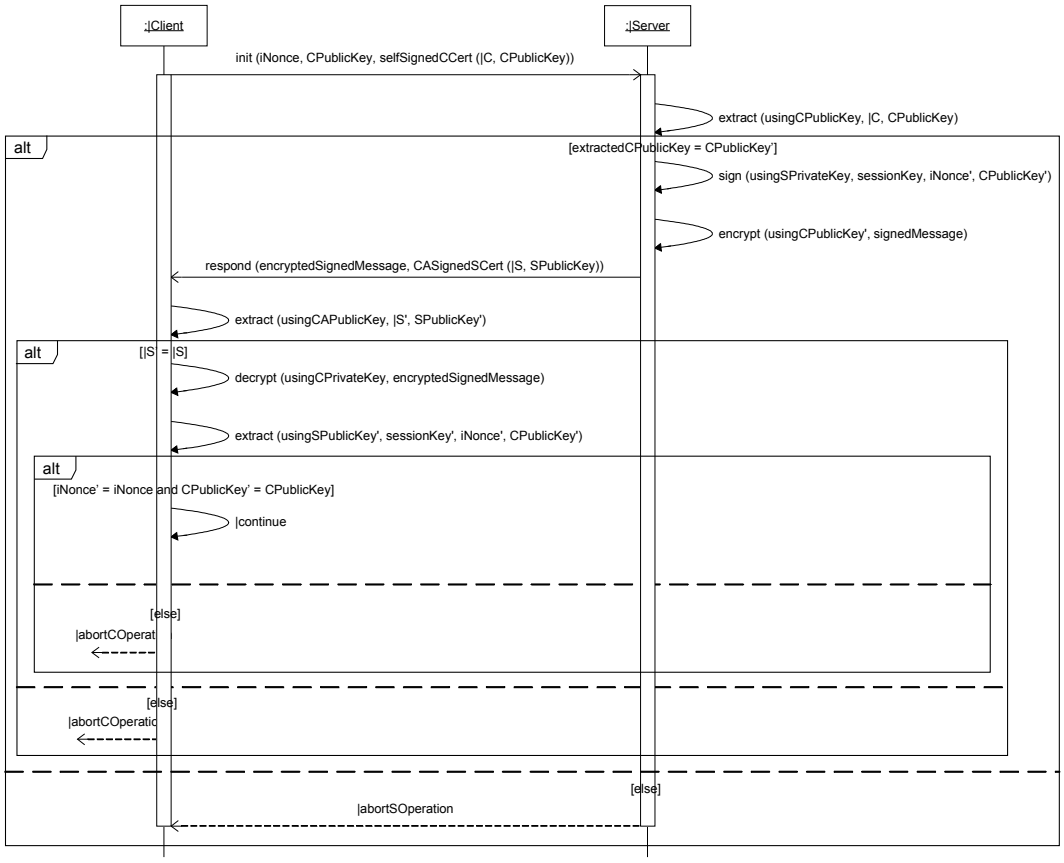

Figure 4. TLS mechanism aspect; interaction template

The TLS aspect model contains two class templates; client and server, which are shown as instances in the sequence diagram in Figure 4. For the purposes of this example, certificate creation and certificate authority public keys are assumed to be obtained in a secure manner. The client must have the certificate authority's public key and the server must have a certificate, signed by the certificate authority (CA), containing its name and public key. Primed variables are used to distinguish between sent and received values. For example, the guard [|S' = |S] tests the received value of |S against the previously sent value of the same variable. Other assumptions include the fact that both nonces (message sequence identifiers) and session keys must change each time the protocol is initiated.

Security verification of the aspect model is performed by transferring the UML diagram in Figure 4 to UMLsec formal semantics, which are expressed in terms of Abstract State Machines (ASMs). These ASMs are built on the UML statechart semantics described by Börger et al. (2000). Since the security aspect is analysed against the adversary behaviour, both the security aspect and the default adversary behaviour are modelled as an ASM adversary machine. The security aspect is executed in the presence of an adversary type *M*. The adversary machine models the actual behaviour of an adversary type *M* as part of the security aspect *a*. This is accomplished with an ASM consisting of two sets of information: (1) the set of control states, $control \quad State$, where *State* is the complete set of states in the ASM and *control* is the set of control states; (2) the set of adversary knowledge $K \quad Exp$, where *Exp* consists of all possible information that an adversary might gain, i.e., all UMLsec stereotyped data in the system specification. In this example we are only interested in whether an adversary can gain the secret data *uname* and *pword*. The UMLsec tool iteratively executes the ASM according to the following schema:

- Specify the initial state $control = control_0$ and the initial adversary knowledge K_0,

- Perform security analysis by checking the data in the link queues and, if the data of any of the link queues (in-queue and out-queue of link *l* connected to the current state) belonging to a link *l* with $read \quad threats_M^S(l)$, where *S* denotes the subsystem being analysed, the data is added to K, else K is unchanged,

- Chose the next control state non-deterministically from the set of control states.

The verification stops when all control states have been visited or if the secret knowledge has been gained, in our example that {*uname*, *pword*}K . Since the UMLsec tool analyses link queues, these must be specified for each message in Figure 4. Message link queues are specified by first placing the content of the message in the out-queue *(outQu)* of the sending object. Next the content of the message is sent on the communication link *l* by removing it from the out-queue of the sender and inserting it into the in-queue (*inQu)* of the receiver. For example, for the *init (...)* message in the TLS aspect, the content of the message is put in the out-queue of the Client, $outQu_C$={iNonce, CPublicKey,selfSinedCCert(|C,CPublicKey)}. The message is sent on the communication link l_1 (since the communication between *client=eCommerceclient* and *server=loginManager* is over the *Internet* as specified in the deployment diagram in Figure 2) by removing the content from the out-queue for the client $outQu_C$ and inserting it into the in-queue of the Server $inQu_S$. Since this message does not contain the secret data *uname* and *pword*, the adversary does not gain the data. Each control state is visited, and each message is processed as described above. None of the messages in the TLS aspect contain the secret data *uname* and *pword*, so this information is never added to K . Thus, the security of the aspect is verified, and the tool gave this result in five seconds.

Security Solution Design Trade-Off Analysis

When choosing among alternative security solutions the decision-maker needs a measurable and relational expression of the security level of the alternative security solutions. The security solution design trade-off analysis supports this by computing and comparing the expected RoSI of the security solutions. RoSI is computed using a set of trade-off parameters, such as priorities, security risk acceptance criteria, standards, laws and regulations, and in particular, business goals, budget, TTM and policies. RoSI for a particular solution is derived by evaluating the effect and cost of the solution against the security requirements, or the misuse impact and frequency, if the solution is intended to treat a misuse. Figure 5 gives an overview of the security solution design trade-off analysis. The parameters on the left side of the figure (security requirement, solution effect, solution cost, misuse cost and misuse impact) are input parameters, meaning the information that is traded off. The parameters on the right side of the figure (security risk acceptance criteria, standards, policies, laws and regulation, priorities, business goals, TTM and budget) are trade-off parameters, meaning the information that is used to trade-off the input parameters. There might be other input and trade-off parameters that are important and should be included, which can be done by tailoring the trade-off analysis, as discussed later in the chapter.

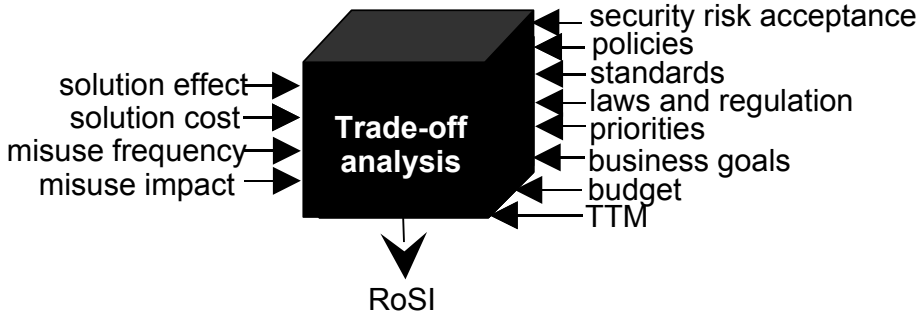

Figure 5. Overview of the trade-off analysis

Potential misuses may reduce the security level of a security solution and are measured in terms of misuse frequency or probability and misuse impact. Misuse impact is given in terms of loss of asset value, and misuse frequency refers to the anticipated number of times within a time period ***p*** that the misuse might occur. Asset is something to which an organisation directly assigns value and, hence, for which the organisation requires protection (AS/NZS 4360, 2004). The value of assets is given in terms of their importance to the business. Security solutions address misuses by either reducing their impact, frequency or both and include mechanisms such as encryption, security protocols, authentication protocols, security extensions to applications and protocols and other similar techniques. Each security solution is characterised by its security properties, its effect and its cost.

Structure of the Trade-Off Analysis

For each security solution, the solution treatment effect *SE* and the solution cost *SC* need to be estimated. This is done using appropriate estimation sets (Houmb & Georg, 2005a). If estimation information does not exist, one may collect and combine such information using different information sources as described by Houmb (2005). The structure of the trade-off analysis is constructed such that it is easy to tailor, change and maintain. The structure supports a step-wise trade-off procedure and is hierarchically constructed. The step-wise trade-off procedure is as follows:

(1) Estimate the input parameters in the set *I*, where *I={MI,MF,SE,SC}* and *MI* is misuse impact, *MF* is misuse frequency, *SE* is security solution effect and *SC* is security solution cost.

(2) Estimate the static security level using information from the development for part 3 of Common Criteria, the security assurance requirements.

(3) Estimate the dynamic security level or the operational security level by computing the risk level, using the prediction model described in Houmb et al. (2005b), based on the parameters *MI* and *MF*.

(4) Estimate the treatment level, using the prediction model in Houmb et al. (2005b), based on the parameters *SE* and *SC*.

(5) Estimate the trade-off parameters in the set *T*, where *T={SAC,POL,STA,LR,BG,TTM,BU}* and *SAC* is security acceptance criteria, *POL* is policies, *STA* is standards, *LR* is law and regulation, *BG* is business goal, *TTM* is time-to-market and *BU* is budget.

(6) Compute RoSI by a) combining the static and dynamic security level, b) evaluating the treatment effect on the combined security level by considering both the security solution effect and the security solution cost and c) Compute RoSI by evaluating the result of a) using the trade-off parameters.

(7) Evaluate RoSI for a particular security solution against the rest of the potential security solutions in the security solution set ***A***.

Figure 6 shows the hierarchical structure of the trade-off analysis. The structure consists of four levels and follows the step-wise description from above. The AND gates in the figure mean that all incoming events linked to the incoming arches are combined. The out-going arches from an AND gate carry the result of the combination to the next level of the analysis. Each of the squares in the figure represents a set of information, for example the set of information that contributes to the static security level.

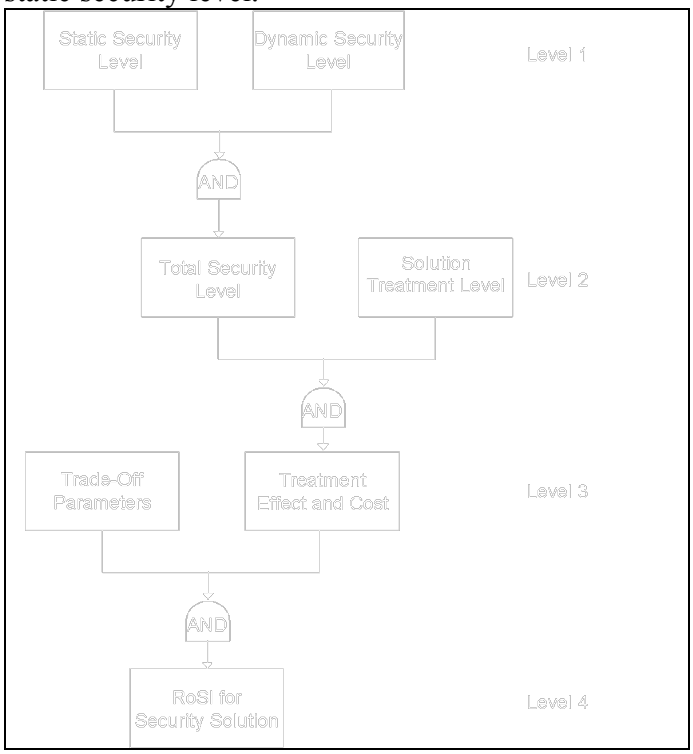

Figure 6: Hierarchical overview of the structure of the trade-off analysis

BBN Implementation of the Trade-Off Analysis

The security solution design trade-off analysis is implemented using Bayesian Belief Networks (BBN). BBN handles large scale conditional probabilities and has proven to be a powerful technique for reasoning under uncertainty. BBN is used for various uncertainty problems, such as support of disease determination based on a set of symptoms in the medical domain, and for the help function in Microsoft Office. It has also been successfully applied when assessing the safety of systems (SERENE Project, 1999).

The BBN methodology is based on Bayes rule and was introduced in the 1980s by Pearl (1988). HUGIN (Hugin Expert A/S, 2004) is the leading tool supporting BBN. Bayes rule calculates conditional probabilities. A BBN is a connected and directed graph consisting of a set of nodes or variables and directed arches. Nodes correspond to events or concepts and are represented in a set of states. The potential states of a node are expressed using probability density functions (pdf). Pieces of information or evidence are inserted into the leaf nodes and propagated through the network using the pdfs. A pdf describes one's confidence in the various outcomes of the node and depends conditionally on the status of the connected nodes. There are three types of nodes: (1) target nodes, which represent targets of the assessment, (2) intermediate nodes, that are nodes for which one has limited information or beliefs (the intermediate level) and (3) observable nodes, which represent information and evidence that can be directly observed (e.g., the number of people on a particular bus at a particular time) or in other ways obtained. Application of the BBN method consists of three tasks: (1) construction of the BBN topology, (2) elicitation of probabilities to nodes and edges and (3) making computations.

Further information on BBN, and in particular on the application of BBN for software safety assessment, can be found in Gran (2002) and the SERENE Project (1999).

The BBN Topology of the Security Solution Design Trade-Off Analysis

Figure 8 gives an overview of the BBN topology of the trade-off analysis. The topology consists of four levels (as llustrated in Figure 6). Here the focus is on the two upper levels of the topology. Evidence and information can be inserted at any level, because the levels are linked together as input and output nodes that carry information between the different levels. In Figure 7 the four nodes *trade-off parameters*, *static security level*, *risk level* and *security solution treatment level* are input nodes (they hide the three other levels), and are indicated by dotted ovals. Each of the input nodes actually consists of additional subnets, but information or evidence can be inserted directly into the input nodes that exist at any particular level.

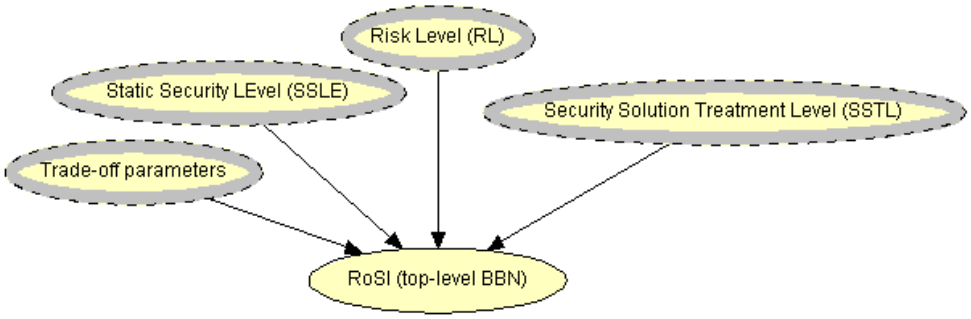

Figure 7: BBN topology for security solution trade-off analysis

Figure 8 shows the top-level net in the BBN topology. The network consists of four main parts: (1) trade-off variables, which are combined by the *trade-off priorities utility*; (2) security level variables, which are combined by the *security level utility*; (3) security risk acceptance variables, which are combined with the security level variables by the *accept risk level utility*; and (4) RoSI variables, which are combined with the variables from the other parts, using the *RoSI utility*. The utility functions (the diamonds in the figure) define the relationship between the nodes on the incoming edges. They are specified using pdfs and can be simple lookup tables or more sophisticated relational expressions (such as if-then-else expressions). The decision variables (the squares in the figure) specify the output from the utility functions. In some cases these variables

work as helper variables, e.g., the security level decision variable and the trade-off priorities decision variable. The security level utility computes the security level based on the static security level variable, risk level variable, and security solution treatment level variable. Each of these variables is computed from its associated subnet. Since the HUGIN propagation algorithm (Jensen, 1996), which is used for the computations, starts with the leaf nodes and propagates one level at the time, the security level and the trade-off priorities are the first to be computed. These two sets of variables are independent and can be computed separately. The next part computed is the acceptable risk level, since it depends on the security level utility, and all other variables depend on this part of the network. Last the RoSI is computed based on the trade-off priorities, acceptable risk level and the variable priorities (PRI). PRI is used as input both when computing the trade-off priorities and when computing RoSI. The reason for this is that the variable are used to determine the outcome of the trade-off priorities utility function, and then used to ensure that those priorities are fulfilled when computing RoSI. PRI represent company or domain-specific issues that must be preserved and, therefore, determines how the other trade-off variables are handled.

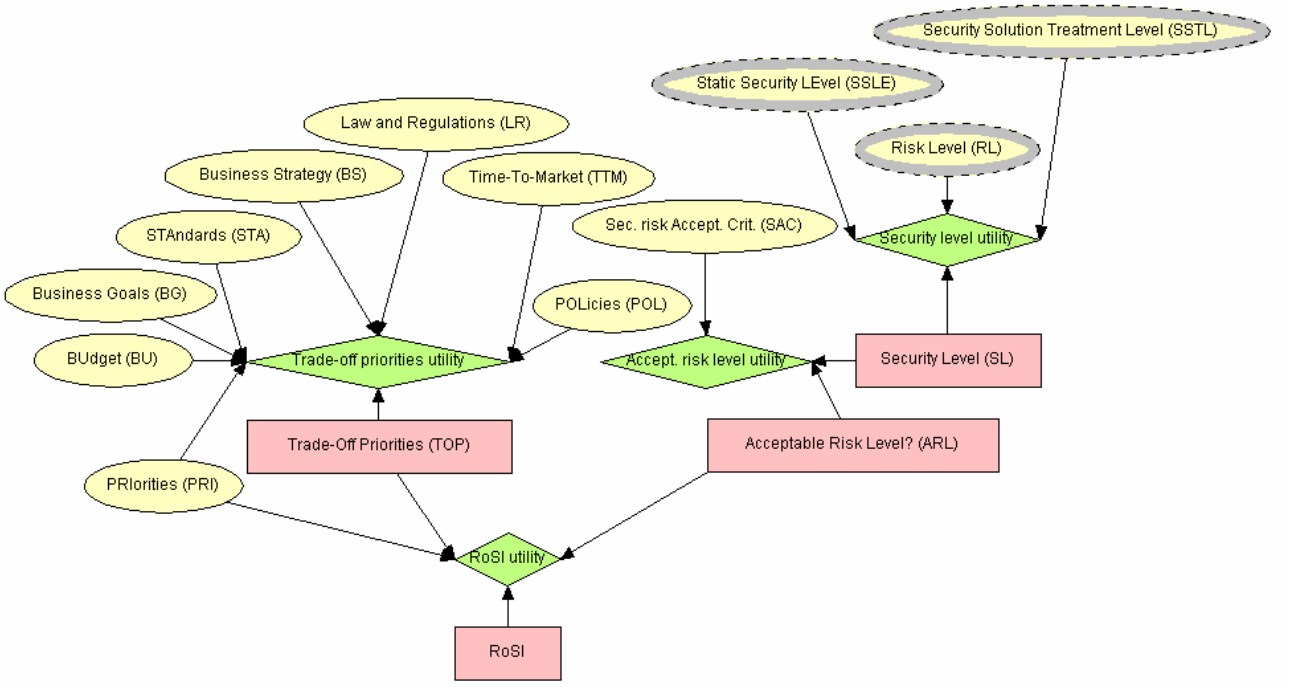

Figure 8: Top-level net in the BBN topology

The target node RoSI represents the objective of the assessment. The trade-off variables each represent different issues that influence which security solution is best for the problem under consideration. The node BU denotes the budget available for mitigating security risks and is given as the interval *[min, max]*, where *min* is the minimum budget available for mitigating security risks (in many cases set to 0) and *max* is the maximum budget available for mitigating risks. The variable BG specifies business goals regarding security issues. BG consists of seven states: confidentiality, integrity, availability, authenticity, accountability, non-repudiation and reliability, according to the security definition of the security standard ISO/IEC 13335: Information technology – Guidelines for management of IT security (ISO/IEC 13335, 2001). The variable STA is used to include standards to which the system must adhere. BS covers security issues related to the business strategy. LR covers laws or regulations for security issues that the system must meet. SAC represents security risk acceptance criteria and specifies acceptable and non-acceptable risks. PRI is used to prioritize the other trade-off variables and makes the BBN topology company, system and domain specific. TTM is given as the interval *[min_date, max_date]*, where *min_date* is the earliest TTM date and *max_date* is the latest TTM date. Table 2 depicts the nodes/variables and states for the top-level BBN.

Table 2. Nodes, states and variables of the top-level BBN

Node/variables	States
RoSI	Conf, Integr, Avail, NonR, Accnt, Auth and Relia

PRI	BU, BG, LR, BS and POL
BU	BU_costlimit
TTM	[min_date,max_date]
BG	Conf, Integr, Avail, NonR, Accnt, Auth and Relia
BS, LR, POL and SAC	Conf, Integr, Avail, NonR, Accnt, Auth and Relia
SL, ARL and TOP	Conf, Integr, Avail, NonR, Accnt, Auth and Relia

Figure 9 shows the static security level subnet, which feeds evidence into the SSLE node in the top-level BBN. This network consists of two parts in three levels: (1) the security assurance requirements of Common Criteria (part 3 of CC) that has three levels and (2) assets and their environment. Part 3 of Common Criteria, the security assurance requirements, describe different general and security specific aspects of a development process and target the evaluation of system against the seven EAL of Common Criteria. Each EAL determines a set of security assurance requirements that needs to be validated by the evaluator during the certification process. We use these criteria rather than part 2 of Common Criteria, the functional security requirements, since such requirements are domain specific and also evolve over time. Certification according to Common Criteria is achieved using the documentation provided during system development and covers the requirement, design and implementation phases. By including the assurance class AMA, maintenance of assurance, the BBN hierarchy also covers the maintenance phase of a system's life-cycle.
The asset and asset environment portion of the BBN is used to specify assets that need to be protected and the related security policies, organisational issues, context in which the assets exist and the stakeholder that either owns or is related to the assets by assigning them a value. A stakeholder is an individual, team, or organisation (or classes thereof) with interest in, or concerns relative to, one or more assets. The context is the strategic or organisational environment in which an asset exists. An organisation is a company, firm, enterprise or association, or other legal entity that has its own function(s) and administrations. Security policy concerns the rules, directives and practices that govern how assets are managed, protected and distributed within an organisation and its systems.
The computation order for this subnet is that the common criteria utility and the asset value utility are computed first, since their variables are independent. The static security level is then computed based on the results from the common criteria and asset value utilities.

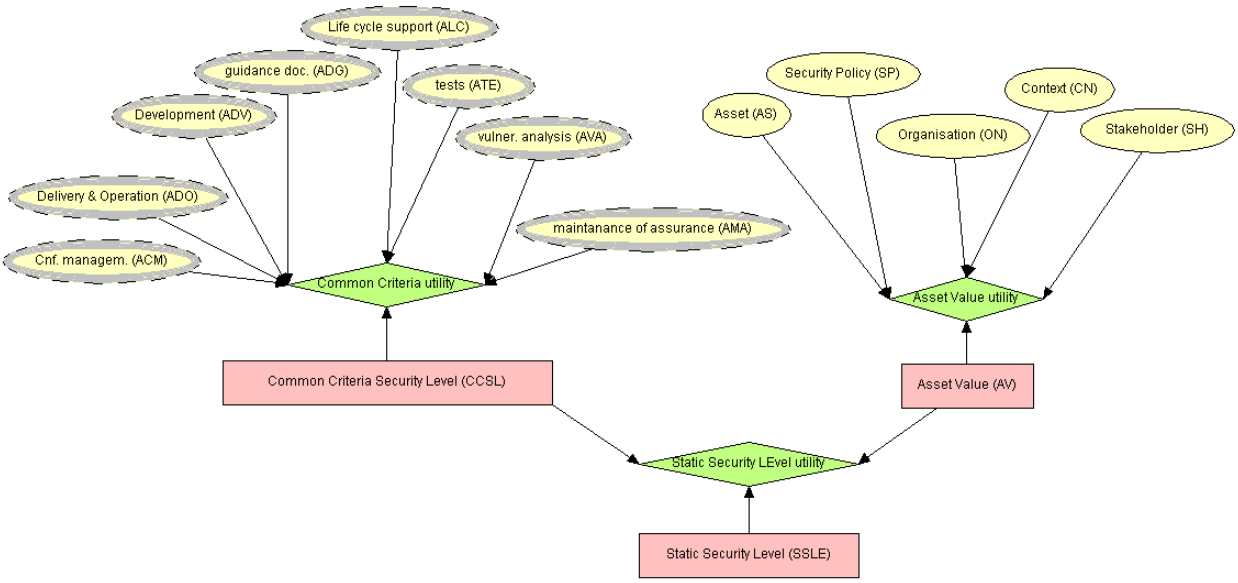

Figure 9: Static security level subnet

Figure 10 shows the risk level subnet. This subnet represents the dynamic security level and feeds evidence into the RL node in the top-level BBN. The subnet consists of three parts in two levels. First the operational risk level and the security risk level are computed. The operational risk level is computed based on the variables MTTM and METM, while the risk level is computed based on the

variables MUSE, MF and MI. Last the risk level is computed using the result of the operation risk level and risk level computations and MC.
Figure 11 shows the security solution treatment level subnet. This subnet feeds input into the SSTL node in the top-level BBN. The treatment level is computed based on the variables SE, SC and SS. SE and SS use the seven security attributes of ISO 13335 as states. The node SC consists of one state, the sc_costlimit, which is given as the interval *[min,max]*, where *min* is the minimum expected cost for the security solution and *max* is the maximum expected cost for the security solution.

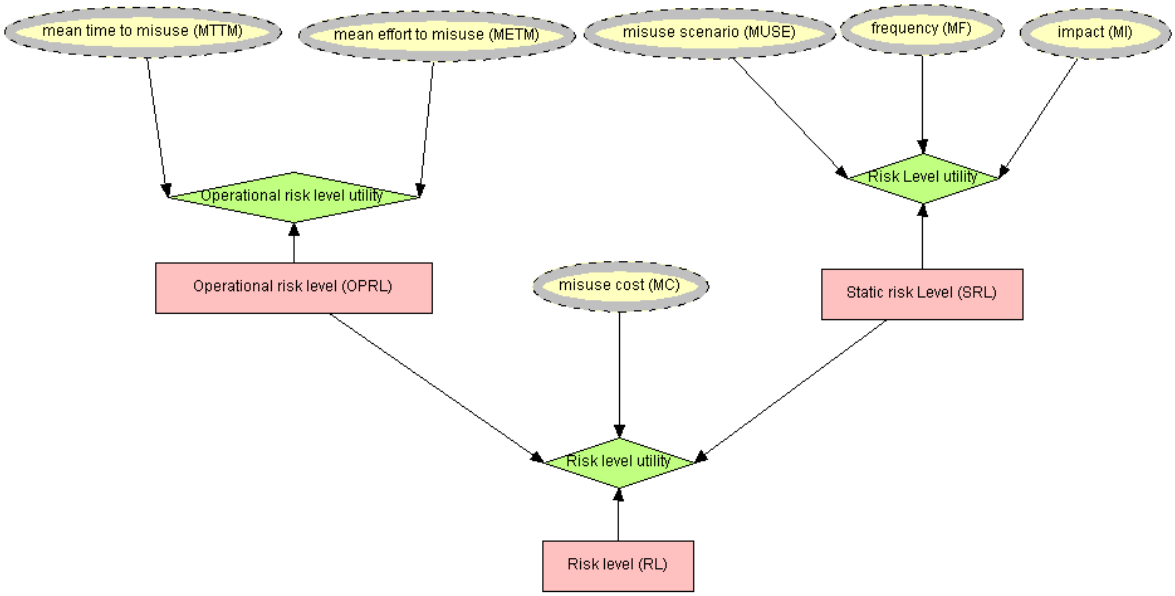

Figure 10: Risk level subnet

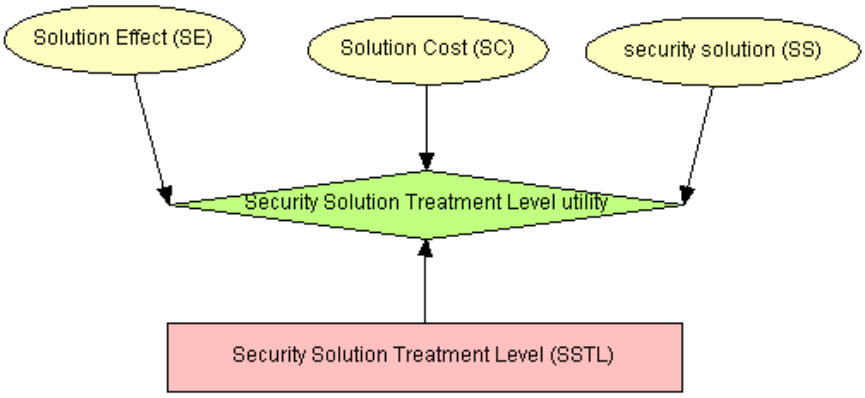

Figure 11: Security solution treatment level subnet

Each subnet has at least one output node that represents the information being transferred to the level above in the BBN topology. Similarly, input nodes represent information given as input from associated subnets. Input nodes are modelled using grey dashed lines. If there are no observations or information available for some of the observable nodes, they are left empty. An example is when the security solution does not target a misuse. In this case the related variables are left empty and not taken into consideration when computing RoSI. Due to space restrictions, we only discuss the security solution treatment level subnet in the example given in the next section.

Demonstration of Security Solution Design Trade-Off Analysis for TLS

Earlier we described the e-Commerce platform ACTIVE, described a misuse scenario and described a potential security solution treating the misuse; a variant of TLS. In this section the SSTL part of the BBN topology is used to demonstrate how to use the security solution trade-off analysis.

Recall that the BBN methodology consists of construction of the BBN topology, elicitation of probabilities to nodes and edges and making computations. The previous section described the BBN topology for the security solution design trade-off analysis. This topology is an implementation of the trade-of analysis and is a general topology for computing RoSI for security solution evaluation. The elicitation of probabilities and computations is, however, domain specific and needs to be assessed in each case.

Before evidence can be entered into the subnet, the utility function for SSTL needs to be defined. Tables 3, 4 and 5 show parts 1, 2 and 3 of the SSTL utility function used in this example. The state category_a of the variable SS refers to the situation when the TLS variant is separate from the web application and works as an add-in to the web application. This means that TLS is not integrated into the application. This also means that the login information is not encrypted between the add-in and the web application, which makes it possible for an attacker to gain the secret information by using directed software sniffers. The state category_b of the variable SS refers to the situation when the TLS variant is integrated into the web application. The different utilities assigned for min and max for the variable SC, related to each of the states of SS, determines the effect of the cost spent on the solution. For example, when the variable SSTL is *high*, the utility for min is set to 0.9. This value of min reflects that not thoroughly checking the implementation (e.g., due to lack of time or money) leads to a small probability that there are mistakes in the implementation, even if the security solution itself is proven to be completely secure and non-modifiable. The reader should treat the utilities given in Tables 3, 4, and 5 as a simple example of how utilities can be used. They do not reflect a general and actual effect and relational specification. Figure 12 shows an example of a computation on the SSTL subnet for the TLS variant when the pdf for *SC* is set to *P(min)=0.7* and *P(max)=0.3*. Figure 13 shows the effect on the computation when the pdf for *SE* is set *to P(Conf)=0.7* and *P(Integr)=0.3*. By inserting evidence in the observable nodes the effects can be observed in the network.

Table 3: SSTL utility function part 1

Variable	**State**							
SSTL	Low							
SS	category_a				category_b			
SC	min		max		min		max	
SE	Conf	Integr	Conf	Integr	Conf	Integr	Conf	Integr
Utility	0.2	0.2	0.4	0.4	0.5	0.5	0.7	0.7

Table 4: SSTL utility function part 2

Variable	**State**							
SSTL	Medium							
SS	category_a				category_b			
SC	min		max		min		max	
SE	Conf	Integr	Conf	Integr	Conf	Integr	Conf	Integr
Utility	0.7	0.7	1.0	1.0	0.7	0.7	1.0	1.0

Table 5: SSTL utility function part 3

Variable	**State**							
SSTL	High							
SS	category_a				category_b			
SC	min		max		min		max	
SE	Conf	Integr	Conf	Integr	Conf	Integr	Conf	Integr
Utility	0.8	0.8	1.0	1.0	0.9	0.9	1.0	1.0

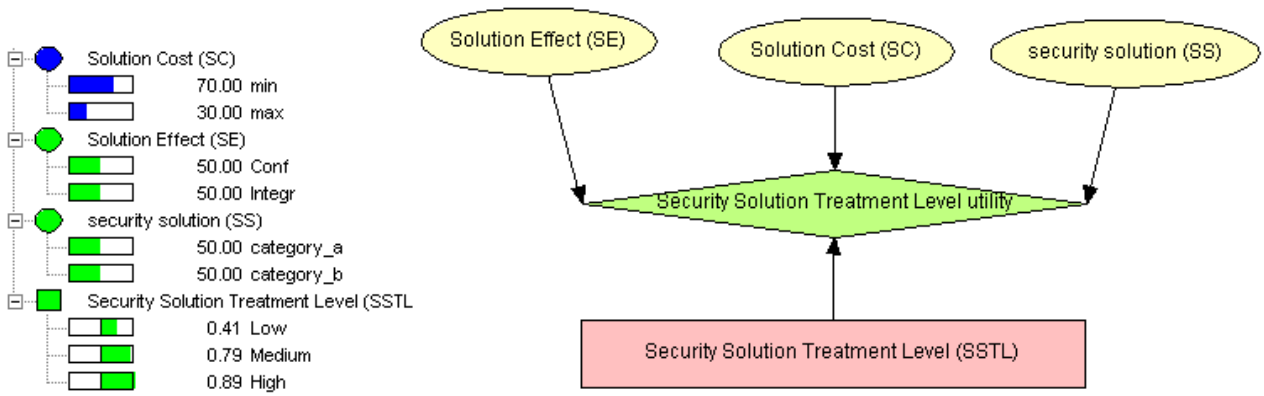

Figure 12: Result of computation when pdf for SC is P(min)=0.7 and P(max)=0.3

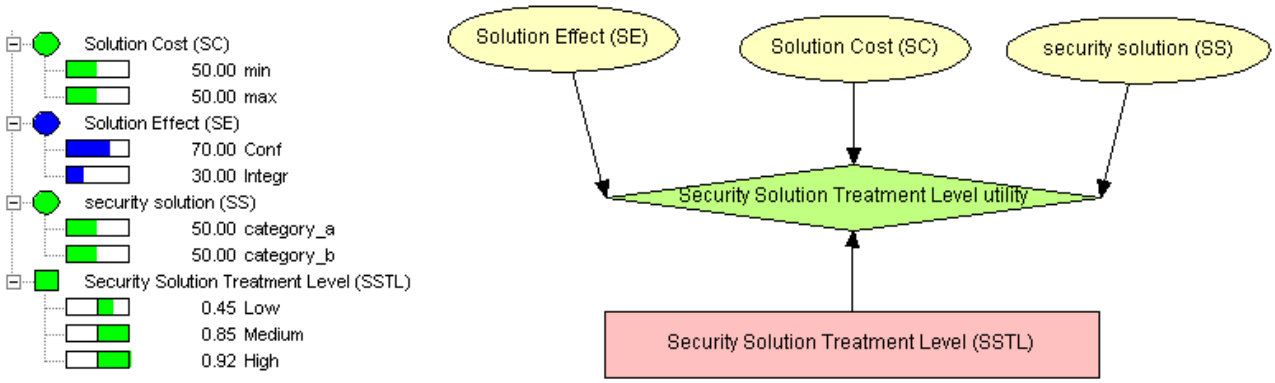

Figure 13: Result of computation when pdf for SE is P(Conf)=0.7 and P(Integr)=0.3

As shown in the SSTL utility functions and by the two example figures, elicitation of probabilities and computation requires information regarding the relation between the different variables as well as values of the variables themselves. The elicitation of probabilities refers to establishing the prior pdf, $P_{prior}(X_i)$, for each state for each node/variable in the network. To compute the effect of information or evidence entered into the network likelihood functions need to be defined, which is done in the utility functions. The utility functions then define the relationship between all sets of states in the network. This is necessary in order to propagate the evidence through the network. Propagate means computations, which is done using the HUGIN propagation algorithm described earlier. The utility functions, or likelihood functions, are used to update the network. This means that the utility functions are a critical part of the network, but it also means that a BBN topology can be tailored for a particular case by changing the utility functions. As argued by both Gran (2002) and SERENE Project (1999), the construction of the network is critical to the outcome of the propagation. One way to aid the construction of the topology and the specification of the utility functions is by using ontologies. We are currently exploring this subject in order to perform topology verification. The current version of the security solution design trade-off analysis is constructed using domain knowledge and is analysed to find the critical parts and the sensitive nodes in the network.

Evidence or information to feed into the network can come from various sources. Houmb (2005) discusses types of information sources and techniques that can be used to combine these sources. There are two main information sources available for estimating the variables in the BBN topology: empirical or observable information and subjective information, such as subjective expert judgments. Observable sources are sources that have directly or indirectly observed the phenomena. Such sources have not been biased by human opinions, meaning that the source has gained knowledge or experience only by observing facts. One type of observable information source is real-time information sources, such as Intrusion Detection Systems (IDS), log-files from Firewalls and honeypots. Other observable information sources are company experience repositories, public experience repositories, domain knowledge, recommendations (best practise) and related standards.

Examples of public repositories are the quarterly reports from Senter for Informasjonssikkerhet (SIS) in Norway, CERT.com, reports from the Honeynet-project and other attack trend reports. Subjective information sources can be direct or indirect, meaning that the expert may have direct or indirect knowledge or experience that he or she uses when offering information or evidence. For example, direct knowledge consists of actual events that the expert has observed. Indirect knowledge refers to events observed by another expert or an observable information source. Here the subjective information source interprets the information given by other sources before providing information. Examples of subjective information sources are subjective expert judgment and expert judgment on prior experience from similar systems. When considering experience from similar systems, the experts need to provide a description of their reasoning about the differences between the systems and the effect of those differences on the configuration being assessed. For more information see Houmb (2005). There might also be situations where there is no information or evidence available. In such situations the symbol $\perp$ is used to indicate lack of information. The variable is still included in the computation, however, to distinguish between no relation, which is denoted *P(event)=0*, and no information available.

FUTURE TRENDS

The techniques described in this chapter compliment the model-driven development (MDD) paradigm. However, for a wide acceptance and application of MDD techniques they must make efficient use of resources and be easy to use across all facets of development. Frameworks that combine MDD techniques and provide tool-support for these techniques must be available to guide the developer in all steps of development of a secure system.

An example of such a framework is the Aspect-Oriented Risk-Driven Development (AORDD) framework (Houmb & Georg, 2005a). The security verification and security solution design trade-off analysis techniques described in this chapter are part of the AORDD framework. The framework consists of an iterative risk-driven development process that uses AOM techniques and two repositories with associated rules that support the BBN implementation of the security solution design trade-off analysis. The repositories store experience for reuse, such as security aspects and their related estimation sets. (Recall that estimation sets include estimations of misuse impacts and frequency, as well as security solution effectiveness and cost.) The rule sets guide annotation of design models (currently UML models) and how information is transferred from those models to the BBN topology. Once the system design is completed, MDD techniques that generate code can be used to realise the system (Jürjens & Houmb, 2005).

Frameworks such as AORDD will help developers unfamiliar with the security domain successfully develop secure systems using the MDD paradigm.

CONCLUSION

The chapter describes an integrated SVDT approach for effective and controlled development of secure systems. SVDT addresses conflicting issues, such as fulfilling a required security level, making effective use of available resources and meeting end-user expectations. Security verification using UMLsec tool-support and security solution design trade-off analysis are two techniques of SVDT. Security verification is used to verify that security requirements are fulfilled by analysing a security solution design that is modelled as a security aspect using UMLsec. The security solution design trade-off analysis is implemented using BBN. The BBN topology consists of four levels that interact through input and output nodes. The topology covers the static security level, risk level, security solution treatment level and trade-off parameters. The topology is organised using subnets to ease the propagation of evidence. Elicitation of probabilities is achieved using available empirical or observable information sources combined with subjective information sources. Probabilities are obtained from experience and stored in the repositories described in the previous section. Computations are performed using the HUGIN propagation algorithm. The trade-off analysis is demonstrated using an example.

The BBN topology used in the security solution design trade-off analysis is a general topology and is not limited to security solution evaluation. However, it is security-specific in its current version,

since the variables used are security-related variables. Tailoring the topology for other types of decision support is possible if other trade-off variables are added. Modifying or changing the security level and risk level nodes and subnets may also be necessary. The topology can also be tailored to a particular security policy or made company-specific by modifying the priorities in the PRI node and by tailoring the TOP, ARL, SL and RoSI utilities in the top-level BBN.
It is important to note that even though the BBN topology automates part of the decision process, it is still merely a representation of the combination of domain knowledge and human interpretation of what is important to take into consideration in such decisions. This means that both a different structure of the BBN topology and different estimation sets will influence the outcome of the computation.

REFERENCES

ACTIVE (2001). EP-27046-ACTIVE, Final Prototype and User Manual. Version 2.0. Deliverable 4.2.2.

AS/NZS 4360 (2004). AS/NZS 4360:2004: Australian/New Zealand Standard for Risk Management. Standards Australia, Strathfield.

Börger, E. & Cavarra, A. & Riccobene, E. (2000). Modeling the dynamics of UML State Machines. In Gurevich, Y. and Kutter, P. and Odersky, M. and Thiele, L., editors, Abstract State Machines: Theory and Applications, vol. 1019 of LNCS, 223-241. Springer.

Clarke, S. (2002). Extending standard UML with model composition semantics. In Science of Computer Programming. 44(1). 71-100. Elsevier.

Clarke, S. & Banaissad, E. (2005). Aspect-oriented analysis and design. Addison-Wesley Professional.

CORAS Project (2005). IST-2000-25031 CORAS: A platform for risk analysis of security critical systems. http://coras.sourceforge.net/. Accessed 28 October 2005.

CORAS Platform (2005). CORAS risk assessment platform, version 2.0. http://sourceforge.net/project/showfiles.php?group_id=88350&package_id=133388&release_id=276903. Accessed 26 September 2005.

Dimitrakos, T. & Ritchie, B. & Raptis, D. & Aagedal, J. O. & den Braber, F. & Stølen, K., & Houmb, S. (2002). Integrating model-based security risk management into Ebusiness systems development: The CORAS approach. In Monteiro, J., Swatman, P., and Tavares, L., editors, Second IFIP Conference on E-Commerce, E-Business, E-Government (I3E 2002), volume 233 of IFIP Conference Proceedings, 159-175. Kluwer.

France, R. B. & Kim, D.-K. & Ghosh, S. & Song, E. (2004a). A UML-based pattern specification technique. In IEEE Transactions on Software Engineering. 30(3). 193-206. IEEE Computer Society.

France, R. B. & Ray, I. & Georg, G. & Ghosh, S. (2004b). An aspect-oriented approach to design modeling. In IEE Proceedings on Software, Special Issue on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, 151(4), 173-185. IEEE Computer Society.

Gran, B.A. (2002). The use of Bayesian Belief Networks for combining disparate sources of information in the safety assessment of software based systems. Doctoral of engineering thesis 2002:35, Department of Mathematical Science, Norwegian University of Science and Technology.

Houmb, S.H. (2005). Combining Disparate Information Sources when Quantifying Operational Security. In Callaos, N., Lesso, W., Hansen, E., editors, Proceeding of 9th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2005), 1:228-235, Orlando, Florida,USA.

Houmb, S. H. & Georg, G. (2005a). The Aspect-Oriented Risk-Driven Development (AORDD) Framework. In Benediktsson, O. et al, editors, Proceedings of the International Conference on Software Development (SWDC/REX), 81-91, Reykjavik, Iceland. Gutenberg.

Houmb S.H. & Georg, G &Reddu, R. & France, R. &Bieman, J. (2005b). Predicting availability of systems using BBN in aspect-oriented risk-driven development (AORDD). 2nd Symposium on Risk Management and Cyber-Informatics (RMCI '05). In Callaos, N., Lesso, W., Hansen, E., editors, Proceeding of 9th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2005), Orlando, Florida,USA.

Houmb, S.H. & Georg, G. & France, R. & Bieman, J. & Jürjens, J (2005c). Cost-Benefit Trade-Off Analysis Using BBN for Aspect-Oriented Risk-Driven Development, Proceedings of 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2005), 185-195, Shanghai, China.

Hugin Expert A/S (2004). BBN-tool HUGIN Explorer™, ver. 6.3. Alborg, Denmark. http://www.hugin.dk. Accessed 24 August 2005.

ISO 15408 (1999). ISO 15408 Common Criteria for Information Technology Security Evaluation. http://www.commoncriteria.org/.

ISO/IEC 13335 (2001). ISO/IEC 13335: Information technology - Guidelines for management of IT Security.

Jacobson, I. (2003a). Case for aspects – Part I. In Software Development Magazine. October. 32-37.

Jacobson, I. (2003b). Case for aspects – Part II. In Software Development Magazine. November. 42-48.

Jensen, F. (1996). An introduction to Bayesian Network. UCL Press. ISBN: 1-85728-332-5.

Jürjens, J. (2004). Secure Systems Development with UML. Springer-Verlag, Berlin Heidelberg, New York. ISBN: 3-540-00701-6.

Jürjens, J. & Houmb, S.H. (2005). Dynamic Secure Aspect Modeling with UML: From Models to Code. In Briand, L. & Williams, C., editors, Proceedings of 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2005), 142-155, Montego Bay, Jamaica.

Jürjens, J. (2005). UMLsec-Webinterface: tool-support for security verification using UMLsec. http://www4.in.tum.de/~umlsec/csduml/interface/. Accessed 8 November 2005.

Kasman, R. & Asundi, J. & Klein, M. (2002). Making Architecture Design Decisions: An Economic Approach. Technical report CMU/SEI-2002-TR-035. http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr035.pdf

Kazman, R. & Klein M. & Clements, P. (2000). ATAM: Method for Architecture Evaluation. Technical report CMU/SEI-2000-TR-004. http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tr004.pdf.

Kiczales, G. & Hilsdale, E. & Hugunin, J. & Kersten, M. & Palm, J. & Griswold, W. (2001). Getting started with AspectJ. In Communications of the ACM. (44)10. 59-65. ACM.

Littlewood, B. & Brocklehurst, S. & Fenton, N. & Mellor, P. & Page, S. & Wright, D. & Dobson, J. & McDermid J. & Gollmann, D. (1993). Towards operational measures of computer security. Journal of Computer Security, 2:211-229.

Object Management Group (2003). OMG MDA guide. OMG. Version 1.0.1. http://www.omg.org

Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems: Network for Plausible Inference. Cambridge University Press. ISBN: 0521773628.

SERENE Project (1999). SERENE: Safety and Risk Evaluation using Bayesian Nets. ESPIRIT Framework IV no 22187. http://www.hugin.dk/serene/. Accessed 30 October 2005.

Stølen, K. & den Braber, F. & Dimitrakos, T. & Fredriksen, R. & Gran, B.A. & Houmb, S.H. & Stamatiou, Y. C. & Aagedal, J. Ø. (2002). Model-based risk assessment in a component-based software engineering process: The CORAS approach to identify security risks. In Barbier, F., editor, Business Component-Based Software Engineering, 189-207. Kluwer. ISBN: 1-4020-7207-4.

Straw, G. & Georg, G. & Song, E. & Ghosh, S. & France, R. & and Bieman, J. (2004). Model composition directives. In Baar, T. & Strohmeier, A. & Moreira, A. & Mellor, S., editors, Proceedings UML 2004, 7th International Conference on UML, volume 3273 of LNCS, 84-97. Springer-Verlag.