

# From Security Modelling to Run-time Security Monitoring

Antti Evesti, Eila Ovaska and Reijo Savola

VTT Technical Research Centre of Finland, Kaitoväylä 1,  
90571 Oulu, Finland  
{Antti.Evesti, Eila.Ovaska, Reijo.Savola}@vtt.fi

**Abstract.** In this paper we take the first steps from security modelling to run-time security monitoring. Providing full support for run-time security monitoring requires that following issues are solved: security concepts has to be defined in an unambiguous way, security level has to be defined and measured, and finally, software has to adapt itself based on measurements and requirements. This paper addresses the unambiguous definition of security by examining existing security ontologies. None of the existing ontologies is able to support run-time security monitoring as such, and there is a need to combine and widen these ontologies. In addition, this paper describes our vision how run-time security management can be achieved as the wholeness.

**Keywords:** Security ontology, security measuring

## 1 Introduction

Today's software products are not running in a static and beforehand known environment. Instead, software is running in mobile devices within constantly evolving environments or, alternatively, software is running in a fixed place but new services become available for exploitation of this device. In addition, the threat landscape is changing constantly. In both cases, the user wants to preserve a particular security level (or security performance) – even though his/her environment changes. In these circumstances, making all security decisions at a design time is not sufficient and thus managing security also at run-time is required. In other words, it is necessary to reveal security changes, by means of monitoring, and adapt the software during its execution – in order to ensure the desired security level for system's users.

Although there are a number of security solutions introduced in the literature [10], [22], there is no common understanding how to define and measure security in practical cases. The existing definitions are generic but too abstract. One example of immaturity of methods and techniques used for security modelling and measurement is the diversity of security concepts and metrics defined in research papers and standards, for example in [1] and [3]. Moreover, security is a cross-cutting issue [2], and therefore, its management is difficult, not only at run-time but also at design-time.

Development of appropriate run-time security solutions is a complex process. Firstly, security has to be defined, i.e. modelled, in an unambiguous way that is

universally understood by all stakeholders. These generic security models, typically represented by means of security ontologies, are used as basis of security aware software development. Secondly, an appropriate security level needs to be defined, measured and monitored constantly. Obviously, a widely-accepted measurement ontology is needed to enable metrics development. Thirdly, software has to be able to adapt itself based on the measurement results. In this paper, we concentrate on the first challenge, security modelling, by examining security ontologies that can be used for representing security. Since these ontologies offer support for security measurement, security measurement issues are also discussed. Finally, we introduce our vision about the run-time security management based on existing security ontologies that are to be combined and enhanced.

The remainder of this paper is organized as follows. Firstly, existing security ontologies are examined, and thereafter, quality and security measurement issues are discussed. Section 3 shows how our approach takes security issues into account at design-time, and, thereafter, we present our vision of the run-time security management. Conclusions and future work section closes the paper.

## 2 Security Ontologies

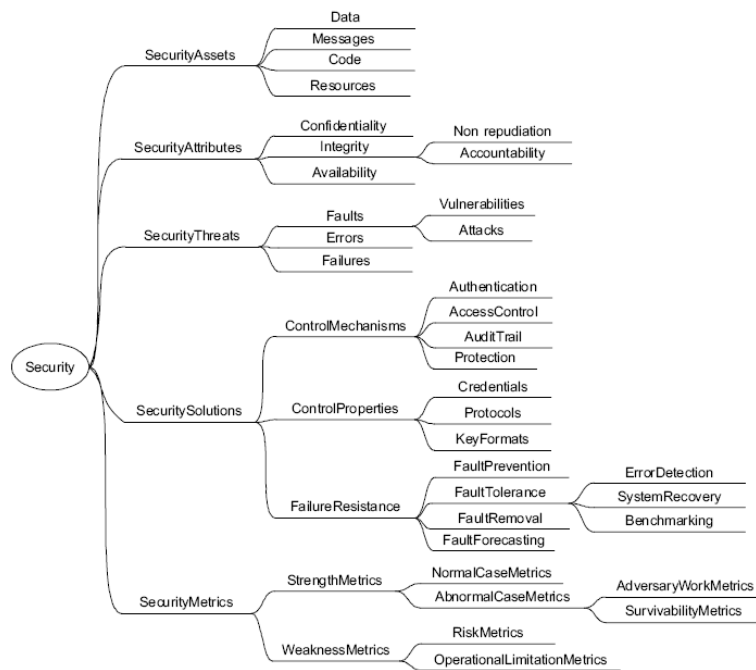
### 2.1 Information Security in General

In order to achieve coherent understanding of security we have to describe security issues in a universal way and incorporate enough details to make the description useful. Ontologies make it possible to give this kind of presentation. We used the following method to select ontologies to this study: 1) ontologies have to concentrate information security, 2) only general purpose ontologies, i.e. not domain specific ontologies, and 3) ontologies have to be mature enough, i.e. at least a concept structure has to be available. Based on these criteria, we describe four existing security ontologies and consider their differences – concentrating applicability to run-time usage and security measurement.

Savolainen et al. present a taxonomy of information security for service centric systems in [7]. The presented taxonomy is intended for the use of software architects of service centric systems. Thus, the taxonomy supports following aspects: 1) stakeholders' participation in the development of service-centric systems, 2) improve communication of security concerns in requirements elicitation, 3) aid to designing and constructing of security means while architecting and 4) support quality analysis phase by providing a common security terminology. [7]

The security taxonomy is divided to five main concepts as shown in **Fig. 1**. *SecurityAssets* contains entities that have a value, i.e. asset means an entity that has to be protected. *SecurityAttributes* contains concepts *Confidentiality*, *Integrity* and *Availability* – also called CIA triad. It must be noted that in telecommunications, this triad is often enhanced with *non-repudiation* and explicit reference to *authentication* and *authorization*. These security attributes compose service's security, and thus, security specification of a system should cover all of these aspects. After that, the concept *SecurityThreats* defines *Faults*, *Errors* and *Failures*. A failure is defined as

an event that occurs when the delivered service differ from the correct service. Instead, deviation in the external state of the system is called an error and the cause of an error is called fault. Internal faults are called vulnerabilities, whereas external faults are called attacks. The *SecuritySolutions* contains means for preventing unwanted behaviour and development of a system. The last concept is *SecurityMetrics* divided to *StrengthMetrics* and *WeaknessMetrics* – containing eight metrics to measure system’s threats and efficiency of their countermeasures [7]. However, this categorization is at very high abstraction level.

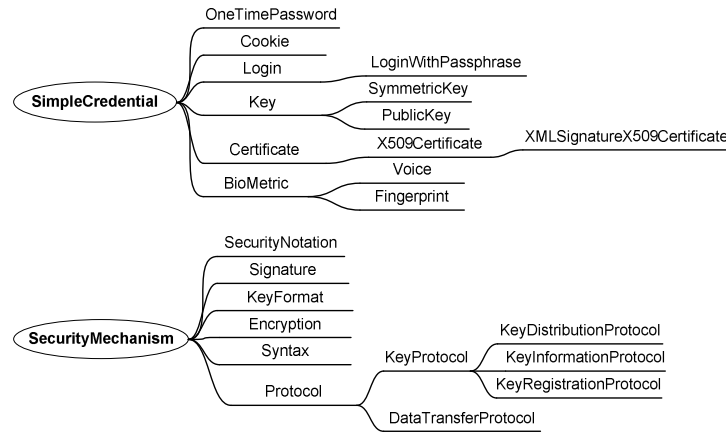


**Fig. 1** Security taxonomy by Savolainen et al.

Denker et al. describe security annotations, represented by DAML-S, intended to be used by agents in [8]. Their ontology can be used to describe service requirements and capabilities – e.g. the requirement that a service requestor wants to use the Open-PGP encryption – and this information are used for service discovery and matchmaking. Matchmaking is performed based on requirements and capabilities with four matching level. [8]

**Fig 2** presents the main parts of ontology from Denker et al. [8]. These ontologies contain *credentials information* and *security mechanisms*. Therefore, many important security aspects are missing. When compared to the work by Savolainen et al., for instance assets, threats and metrics are not defined. However, purpose of the Denker’s ontology is to concentrate mostly to the service discovery and matchmaking, and thus

scope is different than ontology in [7]. Lately, Denker et al. updated their ontology to utilising OWL in [9].



**Fig 2** Security credentials and mechanisms by Denker et al.

Kim et al. presented their security ontology called NRL (Naval Research Laboratory) security ontology in [10]. NRL security ontology makes it possible to annotate resources with security related information that can be used during service discovery and matchmaking. NRL security ontology describes following security aspects: *mechanisms*, *protocols*, *objectives*, *algorithms* and *credentials* in various levels of details. Both service providers and requestors can use NRL ontology to describe their capabilities and requirements. Actually, NRL security ontology is a collection of ontologies. Main Security ontology imports the Credentials, Security Algorithms and Security Assurance ontologies as object properties. Thus, these ontologies make it possible to give more specific values for *SecurityConcepts*. In addition, user can define *SecurityObjectives* and connect them to the specific *SecurityConcept* by utilising *supportSecurityObjective* property. [10]

NRL security ontology is well organised concentrating mostly on security solutions area and providing a reasonable way to matchmaking between requirements and capabilities. When comparing NRL ontology to work by Savolainen et al. it can be noticed that the NRL security ontology lacks in security assets, threats and metric areas. However, NRL security ontology contains a substantially better description in security solutions area. In addition, connections between security objectives (requirements) and their solutions are defined more comprehensively.

Tsoumas et al. present in [11] and [12] a framework for information system security management, i.e. linking high-level policy statements to explicit low-level security controls adaptable and applicable in the information system environment. Authors' framework consists of four phases: 1) Building of Security Ontology, 2) Security Requirements Collection, 3) Security Actions Definition, and 4) Security Actions Deployment and Monitoring. Building of Security Ontology means ontology's instantiation based on a conceptual model. This conceptual model defines:

*Asset, Attack, Controls, Countermeasure, Impact, Security policy, Stakeholder, Risk, Threat, Threat agent, Unwanted incident, Vulnerability* and *relationships* among these concepts. When compared ontology from Tsoumas et al. to ontology from Savolainen et al. it can be noticed that both ontologies describe security in the same abstraction level. Thus, neither describes detailed protocols or mechanism for achieving security as opposed to NRL security ontology does. As well as NRL security ontology and ontology from Denker et al., ontology from Tsoumas et al. suffers from lack of security metrics.

**Table 1** summarizes the most important aspects of above described security ontologies from run-time security management viewpoint. Ontologies from Denker et al. and Kim et al. are initially intended for run-time environment, i.e. service discovery and matchmaking. On the other hand, only ontology from Savolainen et al. contains security metrics. Thus, combining proposals from Savolainen et al. and Kim et al. should offer an appropriate approach. However, combining ontologies in an appropriate way contains many challenges. Firstly, suitable abstraction level has to be found. Secondly, relationships between metrics and security solutions are required, in order to measure solutions' efficiency during the run-time. In addition one of the major challenges is how to incorporate attacker behaviour to the ontology. Malicious activity is the major difference in security compared to other quality attribute descriptions.

**Table 1** Summary of security ontologies

Ontology	Scope	Pros / cons
Savolainen et al.	Design phase of service centric systems.	+ Contains metrics – No connection between attributes and solutions – Only abstract level hierarchy – Intended for design-time use
Denker et al.	Service discovery and matchmaking	+ Run-time applicable – Only credentials and mechanisms.
Kim et al.	Service discovery and matchmaking	+ Run-time applicable + Detailed solutions + Connection between objectives and solutions – Metrics are missing
Tsoumas et al.	Linking policy statements to security controls	– Only abstract level hierarchy – Intended for design-time use

## 2.2 Security Measurement

It is a widely accepted management principle that an activity cannot be managed well if it cannot be measured. Overall, metrics provide four fundamental benefits – to characterize, to evaluate, to predict and to improve. Security metrics and measurements can be used for decision support, especially in assessment and prediction. Examples of using security metrics for assessment include [14]:

- Comparison of different security controls or solutions,
- Security assurance of a product, an organization, or a process,

- Security testing (functional, red team and penetration testing) of a system,
- Certification and evaluation (e.g. based on Common Criteria [15]) of a product or an organization,
- Intrusion detection in a system, and
- Other reactive security solutions such as antivirus software.

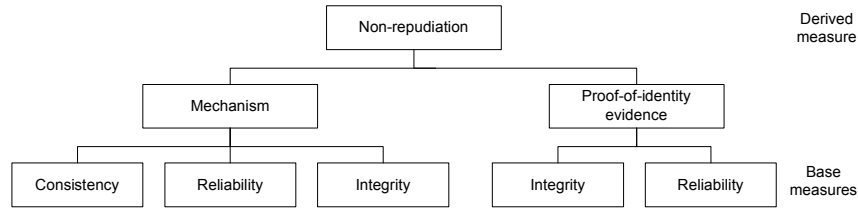
The field of defining security metrics systematically is young and the current practice of information security is still a highly diverse field; holistic and widely accepted approaches are still missing. A major challenge in developing appropriate and feasible security metrics is the immaturity of the state-of-the-art security requirements engineering. Security requirements have not been profoundly addressed within the software engineering community: they are still regarded as being in a side role in most of the software requirements engineering codes of practice [18].

In order to measure, the target of measurement needs to be identified. It is important to clearly know the entity that is the target of measurement because otherwise the actual metrics might not be meaningful. The target of security measurement can be, e.g., an organization, its processes and resources, or a product or its subsystem. The most widely known technical security certification standard is the Common Criteria (CC) ISO/IEC 15408 international standard [15]. During the CC evaluation process, a numerical rating, EAL (Evaluation Assurance Level), is assigned to the target product, with EAL1 being the most basic and EAL7 being the most stringent level. Each EAL corresponds to a collection of assurance requirements, which covers the complete development of a product with a given level of strictness.

ISO/IEC 9126 standard concentrates measuring software's quality, and it contains three metric reports: 1) External metrics [4] applicable during testing, 2) Internal metrics [5] applicable during development and 3) Quality in use metrics [6] applicable in run-time. Furthermore, ISO/IEC divides quality attributes into characteristics and subcharacteristics – security can be found under the functionality characteristic. Therefore, metrics from ISO/IEC are grouped to these characteristics and their subcharacteristics. However, standard does not contain any security related metric in Quality in use metric document [6].

Therefore, it is necessary to find a suitable way to measure security more extensively and holistically. Wang et al. presented a security measurement framework in [2] based on security requirements. The main idea in their work is to divide security requirements to smaller parts, i.e. decomposition, and finally these smallest parts can be measured. Garcia et al. propose two terms: a base measure and a derived measure in [19]. The base measure is a measure of quality attribute that does not depend on other measures, whereas the derived measure is a measure derived from base or derived measures. Therefore, base and derived measure terms can be combined to the approach of Wang et al. **Fig 3** shows the decomposition made for non-repudiation in [2] – combined with the base measure and derived measure terms from [19]. It can be noticed that the lowest level in **Fig 3** contains reliability – and metrics for it can be found, for example from ISO/IEC's standard. Thus, decomposition gives a possibility to find and develop security metrics required for the run-time security monitoring. When searching these metrics following sources can also give a valuable input: The U.S. NIST (National Institute of Information Standards and Technology) SAMATE (Software Assurance Metrics and Tool Evaluation) project [16] that seeks to help answer various questions on software

assurance, tools and metrics. OWASP [17] contains an active discussion and development forum on security metrics. More security metrics are listed in [14] and [18]. Representing these base and derived measures in the ontology also makes it possible to utilise measures at run-time, and in addition, generate new derived measures from the existing measures if needed.



**Fig 3** Decomposition for non-repudiation

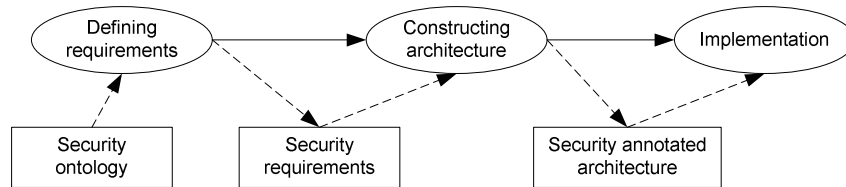
### 3 Security Modelling

This section describes how to design security and how the process should take run-time security management issues into account in order to make it possible to achieve run-time security management.

#### 3.1 Design-time Security Management

Our earlier work has concentrated to take quality issues into account at software’s design and implementation phases by exploiting architectural models [20] and [21], i.e. conforming to model-driven development. We have used ontologies to describe quality attributes in a uniform way. Ontologies are utilized during the quality requirements modelling and during the architecture modelling to select the best solutions to achieve required qualities. Furthermore, we have evaluated a designed architecture in order to detect whether required qualities are met or not. Finally, quality of the implemented software is measured and compared to requirements. All of these phases belong to the QADA (Quality-driven Architecture Design and quality Analysis) methodology [23]. QADA contains two abstraction levels, i.e. conceptual and concrete levels, which can be mapped to the PIM (Platform-Independent Model) and PSM (Platform-Specific Model) from MDA.

Based on our earlier work, we are able to take step forward and start to concentrate to the run-time quality management – especially from security point of view. The means for defining quality attributes (especially reliability) at modelling-time is represented in [20] – and **Fig. 4** follows this approach from the security viewpoint. Thus at this point, the purpose is to model security requirements and the architecture in a way that makes it possible to achieve run-time security management.



**Fig. 4** Main phases of security modelling process.

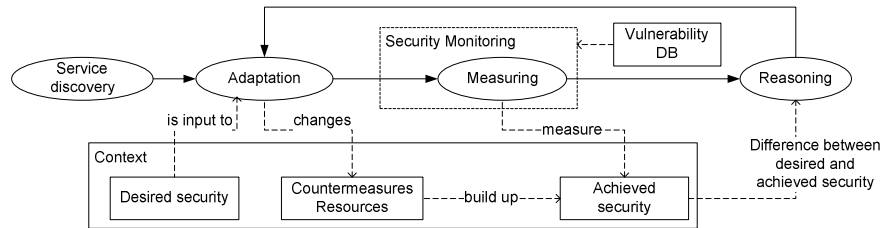
Firstly, ontologies work as an input for the *Defining requirements* activity. Thus, ontologies described in the previous section have to be combined appropriately in order to achieve one adequate ontology – containing at least security objectives, countermeasures, base/derived measures and relationships between these concepts. After that, the security ontology can facilitate a requirements definition activity comprehensively, as a guideline. For instance, an architect can define following security requirements based on security objectives described in the security ontology and the objectives of the designed software: “user has to be authenticated” and “messages’ integrity has to be ensured”.

*Constructing architecture* activity starts after requirements definition. Support for run-time security adaptation requires that alternative security solutions are modelled and implemented into the system. The security ontology helps to find these alternative security solutions. Therefore, the architect selects from the security ontology a password and fingerprints as alternative solutions for the authentication requirement and AES (Advanced Encryption Standard) and DES (Data Encryption Standard) as alternative solutions for the integrity requirement. In addition, the architect selects appropriate measures from the security ontology to monitor achievement of the authentication and integrity requirements. Finally, we have security annotated architecture that works as an input for *Implementation* activity. Therefore, implementation can produce software containing alternative solutions for achieving security on different contexts, and measurements for ensuring that desired security level is kept.

In conclusion, the security ontology is used both defining security requirements and constructing architecture that contains alternative security solutions. Without these steps it is not possible to achieve run-time security management described in the next section.

### 3.2 Run-Time Security Management

Our current vision of the run-time security management is presented in **Fig. 5** – in other words, how the above designed and developed software monitors and adapts its security.



**Fig. 5** Run-time security management

Firstly, available services are discovered. Thereafter, *Adaptation* is performed based on available services, their security properties, and the desired security level. In this phase, the security ontology helps to select the most suitable countermeasure (i.e. security solution as mentioned in the previous section) and resources in order to achieve the desired security level. Furthermore, this phase can utilize the existing service matchmaking solutions, but the adaptation also requires more sophisticated algorithms for the decision making. The purpose is to achieve an automatic adaptation that does not require user actions. However, user preferences can be used to direct a countermeasure selection or setting a divergent desired security level.

Next, the achieved security is *monitored* by means of measuring. A *Measuring* activity measures several properties of the system utilising base measures – introduced in **Fig 3**. From these values the *Monitoring* activity combines an overall security level by means of derived measures. Thus, monitoring activity also requires security ontology. Vulnerability databases can be utilised as an additional input source for monitoring activity. For instance, selected countermeasure might be cracked after ontology construction, and thus its security efficiency is lower than initially set. Hence, utilisation of vulnerability DBs enhances correctness of monitoring.

The achieved security value acts as an input for the *Reasoning* activity, which calls the *Adaptation* if the achieved security level is not satisfying the desired security and alternative countermeasure can be selected. Both the monitoring and the reasoning activities should be automatic – working without user actions. On the other hand, if the desired security level is not achieved and adaptation cannot resolve a situation then user actions will be needed to make a decision how to continue. In **Fig. 5** desired and achieved security levels are part of the context because the security management depends on the context where the adaptation occurs. How context is to be defined is out of the scope of this paper.

As a simple example of run-time security management, software contains AES / DES encryptions and fingerprint / username password pair authentication for achieving its security – as implemented in the previous section. We assume that a user prefers to use fingerprint authentication without encryption as default. In the case when content of the user's information exchange changes, for example, from news reading to more secure online buying, i.e. context changes, the desired security level also changes. The monitoring activity uses base and derived measures from the security ontology for monitoring the achieved security level. Based on the results of the monitoring, the reasoning activity remarks that the desired security level is not

reached anymore. Hence, the adaptation is called, and it decides that the security level for online buying cannot be satisfied without encryption and thus AES is selected – for instance.

## 4 Conclusions and Future Work

Managing security at the run-time requires that: 1) security is understood holistically yet offering enough details to be useful, 2) the achieved security level of the software can be monitored, and 3) software is able to adapt itself based on the context and monitoring results. In this paper, we concentrated to the first issue and also presented the overview vision how the run-time security management can be achieved.

Several security ontologies exist but mostly emphasize service discovery and matchmaking, or alternatively describe different security solutions. Although both are important aspects, these are not enough for covering the whole area of security from the run-time point of view. Thus, there is a need for more extensive security ontology. In addition, security ontology has to contain base and derived measures which make it possible to measure and monitor security levels.

Our next step is to produce the universal security ontology – a combination from the existing ones – containing security objectives and supporting solution mechanisms. Next, we will develop a set of base and derived measures for measuring efficiency of security solutions and include these metrics to the combined ontology. After that we are able to move forward to research monitoring mechanisms and adaptation algorithms required to support the whole process. When the first monitoring mechanisms are available we can build up an initial laboratory case to test validity of the approach and reveal it costs related the performance and other quality attributes.

**Acknowledgments.** This work is made under SOFIA (Smart Objects For Intelligent Applications) project – funded by Tekes (the Finnish Funding Agency for Technology and Innovation) and the European Commission.

## References

1. Avižienis, A., Laprie, J-C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing, Volume 1, Issue 1, pp. 11-33 (2004)
2. Wang, C., Wulf, W. A.: Towards a framework for security measurement. 20<sup>th</sup> National Information Systems Security Conference, Baltimore, pp. 522-533 (1997)
3. ISO/IEC: 9126-1 Software engineering – Product quality – Part 1: Quality model (2001)
4. ISO/IEC: 9126-2 Software engineering – Product quality – Part 2: External metrics (2003)
5. ISO/IEC: 9126-3 Software engineering – Product quality – Part 3: Internal metrics (2003)
6. ISO/IEC: 9126-4 Software engineering – Product quality – Part 4: Quality in use metrics (2004)

7. Savolainen, P., Niemelä, E., Savola, R.: A Taxonomy of Information Security for Service-Centric Systems. 33<sup>rd</sup> EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 5--12 (2007)
8. Denker, G., Kagal, L., Finin, T., Paulucci, M., Sycara, K.: Security for DAML web services: Annotating and matchmaking. In Proc. of the 2<sup>nd</sup> International Semantic Web Conference (ISWC2003), Sanibel Island, Florida, pp. 335-350 (2003)
9. Denker, G., Kagal, L., Finin, T.: Security in the Semantic Web Using OWL. Information Security Technical Report, Volume 10, Issue 1, pp. 51-58 (2005)
10. Kim, A., Luo, J., Kang, M.: Security Ontology for Annotating Resources. OTM Confederated International Conferences, CoopIS, DOA, and ODBASE. Springer, Heidelberg, pp. 1483--1499 (2005)
11. Tsoumas, B., Dritsas, S., Gritzalis, D.: An Ontology-Based Approach to Information Systems Security Management. In Computer Network Security, pp. 151-164 (2005)
12. Tsoumas, B., Gritzalis, D.: Towards an Ontology-Based Security Management. 20<sup>th</sup> International Conference on Advanced Information Networking and Applications AINA), pp. 985-992 (2006)
14. Savola, R.: Towards a Taxonomy for Information Security Metrics. In Proc. of ACM Workshop of Quality of Protection (QoP'07), Alexandria, Virginia, USA, pp. 28-30 (2007)
15. ISO/IEC: 15408-1 Common Criteria for Information Technology Security Evaluation – Part 1: Introduction and General Model. (2005)
16. Black, P. E.: SAMATE's Contribution to Information Assurance. IANewsletter, Volume 9, Issue 2 (2006)
17. OWASP: Open Web Application Security Project. <http://www.owasp.org/>
18. Savola, R., Abie, H.: Identification of Basic Measurable Security Components for a Distributed Messaging System. 3rd Int. Conf. on Emerging Security Information, Systems and Technologies (SECURWARE), Jun 18-23, 2009, Athens, Greece (2009) in press
19. Garcia, F., Bertoa, M. F., Calero, C., Vallecillo, A., Ruíz, F., Genero M.: Towards a consistent terminology for software measurement. In Information and Software Technology, Volume 48, Issue 8, pp. 631-644 (2006)
20. Niemelä, E., Evesti, A., Savolainen, P.: Modeling Quality Attribute Variability: 3<sup>rd</sup> international conference on Evaluation of Novel Approaches to Software Engineering (ENASE), pp. 169-176 (2008)
21. Evesti, A., Niemelä, E., Henttonen, K., Palviainen, M.: A Tool Chain for Quality-driven Software Architecting. 12<sup>th</sup> International Software Product Line Conference (SPLC), p. 360 (2008)
22. Anderson, R.: Security Engineering – A Guide to Building Dependable Distributed Systems. John Wiley & Sons, New York (2001)
23. QADA (Quality-driven Architecture Design and quality Analysis). [www.vtt.fi/qada/](http://www.vtt.fi/qada/)