

Network Security Project Management: A Security Policy-based Approach

Jihene Krichene and Noureddine Boudriga

Abstract—Managing security projects is a delicate activity due to the evolution of attacks. In this paper, we develop a new methodology for estimating security effort based on algebraic representation of security policies. This methodology is used within the SECOMO model. Two models are defined: the *a priori* model and the *a posteriori* model. Real security projects are used to prove the accuracy of the new methodology.

I. INTRODUCTION

Project management allows organizations to conduct their affairs in optimal ways. As any project, security projects have to be managed to ensure they progress correctly. Many methodologies and models have been developed since many years (e.g., COCOMO). All of them use the software size as a basis for estimating the effort and the cost of the project. As security project management is concerned, we notice that the network size is of great importance to estimate security effort as it has been shown in the SEcurity COst Model (SECOMO) that we have developed in [1].

In this paper, we enhance the SECOMO model by giving a new methodology for estimating telecommunication network size. In this methodology, we consider the security policy (SP) as a specification of the security solutions. In fact, SPs have several similarities with software specifications as their rules give instructions for securing a communication system. Similarities existing between specifications and SPs do not deny the existence of differences such as the evolutionary environment for which the SP must be elaborated. The Evolutionary character is mainly due to the emergence of new attacks, vulnerabilities, and technologies.

Since SPs can be considered as specifications, we estimate security effort based on the SP size. For the sake of clarity and need for reasoning, we use the algebraic representation of SPs as it presents the advantage of giving a uniform representation of any SP. Based on the algebraic representation of the SP, its size is estimated using the VSM technique. Statistics are collected each time a security project is conducted and managed using the SECOMO model and VSM size estimation technique to empirically validate the estimation method.

This paper is organized as follows. Section II gives overview about the SECOMO model. Section III presents several techniques used to estimate software size, discusses the technique that has been used within SECOMO in an earlier research and gives a new methodology for estimating the network size based on the algebraic representation of SPs

and the VSM technique. Section IV gives the SECOMO *a priori* model and empirically validates the new methodology using a few real security projects. These statistics are used to refine SECOMO equations, define the *a posteriori* model and adjust it. Finally, Section V concludes this paper.

II. SECOMO: A MODEL FOR MANAGING SECURITY PROJECTS

We have addressed security projects management in a previous work and we have developed in [1] a **Security Cost Model** (SECOMO) which provides security teams with a solid basis for determining how much time, cost and personnel are required for security projects. SECOMO development has been founded on the **Constructive Cost Model** [2], [3] (COCOMO II version) due to the similarity existing between the security engineering management and the software engineering management. SECOMO aims at estimating the effort required to conduct a security project. The effort estimation serves as the basis of other tasks such as the project's duration, number of persons that will carry out the project, and project's cost. These estimations are performed using the concept of network size and various parameters, called scale factors and effort multipliers, that give a measure of the security task complexity. Effort estimation, expressed in *man * time - unit* is given by:

$$E = a \times EAF \times S^b \quad (1)$$

where a is a constant, EAF is an Effort Adjustment Factor, $EAF = \prod_i EM_i$, the EM_i 's are the Effort Multipliers, S is the size of the network, and $b = \beta + \sum W_i$, where β is a constant and the W_i 's are the scale factors. Equation (1) is similar to the effort equation developed for the effort estimation in COCOMO. SECOMO uses four scale factors and thirteen effort multipliers classified into four categories. Despite the similarity existing between SECOMO and COCOMO in using both effort multipliers (EM) and scale factors (SF), we highlight the fact that these parameters are different in the two models. In fact, EM and SF must be appropriate to the project's field; and it is clear that security and software fields present large differences. For instance, in SECOMO model we use an effort multipliers category that is totally appropriate to security projects called "*Information System Factors*" in which we consider audit and attack frequencies since they have an impact on the effort required in securing a network. Even if some EM or SF maintain the same name as in COCOMO, their description is appropriate to the security field. For example, the SITE effort multiplier is considered in COCOMO in the case where

Communication Networks and Security (CN&S), Engineering School of Communications (SupCom), University of 7th November at Carthage, Tunisia. jkrichene@gmail.com nab@supcom.rnu.tn

team members are not fully collocated which infers the use of interactive multimedia for communication between the developers. However, this effort multiplier is used in SECOMO to reflect the geographical distribution of the systems to be secured since distributed systems need more security considerations than centralized systems.

The duration D of a security project is a function of the effort estimated E . It is determined as follows:

$$D = c \times E^d \quad (2)$$

where c is a constant, E is the effort estimated previously, and $d = \delta + \sum W_i$ where δ is another constant. For the effort equation as well as for the duration equation, a , β , c and δ are constants used to calibrate the model. The number of persons involved in the security project is therefore equal to the ratio between the effort and the duration as illustrated by the equation: $P = \frac{E}{D}$. The total cost of the security project is then estimated using the equation $C = MS \times P$, where MS is the Mean Salary of the team members. Network size (S) is the most complex parameter in this model because of the heterogeneity of elements within telecommunication networks. Hardware and software components must be considered without omitting the personnel working on them. Moreover, interaction between system elements makes it more challenging the estimation of the network complexity.

III. A TECHNIQUE FOR NETWORK SIZE ESTIMATION

Security effort estimation is based on the size of the network to be secured. The question is “how is the network size estimated?” Before answering this question, let us discuss existing techniques in the case of software.

A. Software size estimation

Software size estimation has been addressed by several researchers since earliest years in the last century. Many techniques have been used to estimate software size. Samples given here are reduced to Source Lines of Code, Function Point Analysis, and Vector Size Measure.

a) Source Lines of Code (SLOC): Code size is expressed in thousands of source lines of code (SLOC) [2]. A SLOC may involve the test plans and documentation if they are developed with the same care as the delivered software. The goal is to measure the amount of intellectual work put into program development. However, defining a line of code is difficult because of conceptual differences involved in accounting for executable statements and data declarations in different languages.

b) Function Point Analysis (FPA): This technique measures a software project by quantifying the information processing functionality associated with major external data or control input, output or file types [4]. The resulting numbers (Unadjusted FP) are generally converted to lines of code as in COCOMO II [2].

c) Vector Size Measure (VSM): The VSM technique measures the software size using both functionality and problem complexity in a balanced and orthogonal manner [5]. The software size is then seen as a two-dimensional vector which has both magnitude and direction. In [5], magnitude is the software size which is the primary input for effort estimation. To overcome the problem of existing software size measures in addressing adequately the problem complexity of software systems, Hasting and Sajeew derivate software attributes (i.e. functionality and problem complexity) from Abstract Data Types (ADT) specifying the software. If f_A and c_A are respectively the functionality and problem complexity of an ADT, then the size of this ADT S_A is defined by the tuple (f_A, c_A) . The size of the software system is given by the sum of all the abstract data types within the software specification. The magnitude m of an ADT is defined as $m = \sqrt{f_A^2 + c_A^2}$.

B. Telecommunication network size estimation

As telecommunication network size estimation is concerned, we highlight that this issue has not been addressed a lot. In this section, we discuss a network size estimation technique that we have proposed in [1], then we propose a new and more accurate estimation, since inaccuracy has occurred in [1] for the following reasons:

- 1) The proposed technique is based on the network components such as hardware and software components. This can bring us to under or over estimates. In fact, we can implement several applications on a single machine, we can also distribute them on more than one machine. As estimated with this technique, the size will not be the same in the two cases. However, the complexity has not changed.
- 2) Interaction between network elements is not considered by this technique. However, this factor has great effect on the network complexity, which increases as the network components are more related to each others. For instance, the complexity of a system will increase if it uses a public key infrastructure for security purposes.

It comes without saying that network size must be estimated accurately. Neither over nor under estimates are tolerated. As accurate estimation is required, we need a complete specification of the system. Any omitted aspect will cause a false estimation. For this reason, we refer to the security policy where a full specification of the network is provided. In fact, the SP is the unique document which has the advantage of specifying the system architecture, and the various interconnections between internal components as well as those with the external environment to the system. In addition, security countermeasures, that make more complex the system, are specified in this document. Thus, to estimate the size of a telecommunication network, we use an approach based on the SP's ADT size.

It is then clear to the reader that an accurate network size estimation is based on the algebraic specification of the SP as it (1) gives a full description of the network and (2) formally

describes the network in concise, consistent and complete manner. The question is now “which estimation technique will be used?”. To answer this question, we discuss some techniques that are close to those stated above in this section.

The simplest way will be to count the lines within the algebraic specification of the SP. A line will be any sort declaration, any operation or predicate declaration and any axiom definition. Despite its simplicity, this technique is not the most appropriate one if we look for accuracy. In fact, the estimated size will be great less than the actual one. This can be explained by the fact that declared operations and predicates do not present the same number of parameters and thus have not the same complexity. Moreover, the axioms may be more or less complex, depends on the rule they express. Some rules are very complex and take several operands and operators ($\forall, \vee, \wedge, \Rightarrow$) in their specification.

As operators and operands are of great importance, we propose to use the VSM technique for measuring network size as it considers in addition to the functionality, the complexity of what is measured. Using this technique, we measure the functionality of the SP which is shown in the *operation* and *predicate* parts of the algebraic specification. However, the complexity is measured based on the *axiom* part of the specification.

C. Algebraic specification of security policies

Let us highlight that the term security policy refers to numerous aspects of information systems’ security such as network SP, access control SP, key management SP, etc [6]. To unify all these views, we define a security policy as “a set of rules that determine how a particular set of assets should be secured”. Furthermore, the SP should be split into multiple components as it should address all the security requirements of the enterprize. RFC 2196 defined a list including the major components of a SP. We found that all of these policy components can be specified similarly even though they use distinguished security techniques. Abstracting away from its context, a SP representation should contain the protected asset, the operations modeling their interaction, and the security properties that must be followed. Algebraic specifications allow to build an abstraction reasoning about the SP. Many-sorted signatures [7], [8] can be used to handle resources and operations, while conditional axioms can model the security requirements. According to this view, the assets of the protected infrastructure are categorized into sorts. For example, to establish a connection between two machines, we need three sorts (i.e. host, port, protocol). For more details about algebraic preliminaries we refer the reader to [9].

To illustrate what has been stated, we consider a network consisting of some PCs connected to the public network. The access to objects on every PC is secured using the Bell-LaPadula (BLP) multi-level access control model. The BLP model restricts access to objects based on the sensitivity of the information contained in the objects and the formal authorization of subjects to access information of such sensitivity. It also supports discretionary access control

by checking access rights from an access matrix. In this model, an access request (subj, obj, acc) is granted if and only if all of the following properties are satisfied:

- 1) simple security property (no read up): if acc is read, then level(subj) should dominate level(obj).
- 2) *-property (no write down): if acc = append, then level(obj) should dominate level(subj); if acc = write, then level(obj) should be equal to level(subj).
- 3) discretionary security property: the (subj, obj) cell in the matrix contains acc.

Let *unclassified*, *confidential*, *secret* and *topsecret* be four classification levels such as *unclassified* < *confidential* < *secret* < *topsecret*. The formal specification of the network and of the two first properties of the BLP model are respectively given in Fig. 1 and Fig. 2.

All the operations defined in this signature correspond to constants except *hasl*, which is an application assigning a clearance to a subject or an object. The predicate *req* represents an access request for a *subject* to an *object* according to an *accessmode*. Axioms Φ_{1-3} formally model the properties (1) and (2) discussed above.

D. Security policy size estimation

As we have mentioned earlier in this section, we use the VSM technique to measure the SP size based on its algebraic specification. This section shows how we measure the functionality and the complexity of a specified SP and how to derive the size of the policy using these parameters. Illustrative examples are given based on the specifications provided by Fig. 1 and Fig. 2.

1) *SP functionality*: According to Fenton [10], [11], product size is influenced by several metrics including functionality. The causal model for software resources he has established shows that the product size is correlated with the functionality parameter. In addition, to its independence from programming language, functionality has the advantage of counting the amount of functioning rather than the physical length allowing thus better estimation of the product size at early phases. Poels states in [12] that “specification products that represent domain entities and that have a state that may change as a result of business functioning (i.e. through the participation in business events) are characterized by the functionality attribute”.

Like software specification, SPs represent network elements and the different functions that manipulate them. It is then obvious that functionality is an important attribute helping measuring the SP size. Functionality characterizes the information system to be secured and is deduced from

$pres \Pi_1$	<i>Sorts</i>	<i>IPs</i>
	<i>Opns</i>	193.95.10.10, 192.168.100.1, 192.168.100.2, 192.168.100.10, 192.168.100.11, 192.168.100.12, 192.168.100.13: \rightarrow IPs

Fig. 1. Network architecture algebraic specification

<i>pres</i> Π_2	<i>Sorts</i>	<i>actions, objects, subjects</i> <i>access, levels</i>
	<i>Opns</i>	<i>grant</i> \rightarrow <i>actions</i> <i>read, append, write</i> \rightarrow <i>access</i> <i>unclassified, confidential,</i> <i>secret, topsecret</i> \rightarrow <i>levels</i> <i>hasl</i> $: objects \cup subjects \rightarrow levels$
	<i>Preds</i>	<i>req</i> $: subjects \times objects \times access$
	<i>Axioms</i>	$\Phi_1 \forall s : subjects, o : objects.$ $(hasl(s) > hasl(o)) \wedge$ $req(s, o, read) \Rightarrow grant$ $\Phi_2 \forall s : subjects, o : objects.$ $(hasl(s) < hasl(o)) \wedge$ $req(s, o, append) \Rightarrow grant$ $\Phi_3 \forall s : subjects, o : objects.$ $(hasl(s) = hasl(o)) \wedge$ $req(s, o, write) \Rightarrow grant$

Fig. 2. BLP algebraic specification

the SP specifying this system. We define the functionality of a many-sorted signature $\Sigma = \langle S, \Omega, \Pi \rangle$ as the distinct number of non-redundant syntactic properties measured by the sum of atomic units (AUs) in the operation and predicate sections of the specification. The following rules apply:

- 1) A syntactic property is composed of one function or predicate identifier plus zero or more parameters.
- 2) In the case of constant declaration, if more than one constant belong to the same sort, the latter is considered just once.
- 3) The addition of one atomic unit, that defines a syntactic property, increases the functionality of a many-sorted signature by exactly one AU.
- 4) The removal of one atomic unit, that defines a syntactic property, reduces the functionality of a many-sorted signature by exactly one AU.
- 5) A many-sorted signature with no syntactic properties has zero functionality.
- 6) A many-sorted signature with exactly one void function (that does not take any parameter nor return any value) has unit functionality.

Thus, the functionality of a many-sorted signature, Σ , is:

$$f_{\Sigma} = \sum AU \quad (3)$$

For instance, the BLP signature specified in Fig. 2 has five function signatures (three constant declarations, one operation, and one predicate declaration) where the *actions, access, levels* constants have respectively two, four, and five functionality units, the operation *hasl* and the predicate *req* has four functionality units each. Thus, the functionality of the BLP signature is: $F_{BLP} = 19$.

2) *SP complexity*: Problem complexity is a second attribute that influences product size as shown by Fenton [10], [11] and Poels [12]. The latter states that “*the complexity of a domain is a function of the business rules*”. In our case, we deal with security rules that have to be respected to guarantee

an accepted security level and that are stated in the SP. The complexity of the SP is then measured using security rules specified in the axiom section of the algebraic specification.

The problem complexity of a many-sorted signature $\Sigma = \langle S, \Omega, \Pi \rangle$ is defined as the distinct number of non-redundant semantic properties measured as the sum of atomic units in the semantic section of the specification. Semantic properties are specified by axioms. The following rules apply:

- 1) An axiom is specified as a predicate list. A predicate list may be a simple expression or a compound expression. Expressions are recursively defined in terms of atomic units.
- 2) The addition of one atomic unit that defines a semantic property increases the problem complexity of a many-sorted signature by exactly one AU.
- 3) The removal of one atomic unit that defines a semantic property reduces the problem complexity of a many-sorted signature by exactly one AU.
- 4) A many-sorted signature with no semantic properties has zero problem complexity.
- 5) A many-sorted signature with exactly one atom, e.g., $v = true$, has unit problem complexity.

Thus, the complexity of a many-sorted signature, Σ , is:

$$c_{\Sigma} = \sum AU \quad (4)$$

For instance, the BLP signature specified in Fig. 2 has three axioms where Φ_1 , Φ_2 and Φ_3 have nineteen problem complexity unit each. Therefore, the problem complexity of the BLP signature is: $c_{BLP} = 57$.

3) *SP size*: According to Fenton, the software product size is function of functionality, problem complexity and length parameters. However, he did not stated any equation for measuring the software size. An advanced research elaborated by Hastings and Sajeev gives a formula for measuring the software size using the functionality and problem complexity parameters. The accuracy of this formula has been proved by a case study of eight projects.

Similarly to the software field, we measure the network size using functionality and complexity of the SP. For this we use the formula stated by Hastings and Sajeev. The size of a SP is then defined as the sum of the size of the signatures specified within the measured policy. The size of a many-sorted signature S_{Σ} is equal to the magnitude of the Vector Size Measure (VSM) defined by Hastings and Sajeev in [5]. Therefore, S_{Σ} is defined by:

$$S_{\Sigma} = \sqrt{f_{\Sigma}^2 + c_{\Sigma}^2} \quad (5)$$

For instance, the size of the BLP signature is equal to $S_{BLP} = \sqrt{19^2 + 57^2} = \sqrt{3610}$. The size of the specified SP is then equal to $\sqrt{3610} + 8$ where 8 is the size of the network architecture signature.

IV. SECOMO INITIALIZATION AND ADJUSTMENT

In this section we initialize the SECOMO *a priori* model and we prove the accuracy of the proposed technique through

a few security projects. Collected security statistics are used to define and adjust the SECOMO *a posteriori* model.

A. The SECOMO *a priori* model

SECOMO *a priori* model has been defined through the management of three security projects that have been performed in middle sized tunisian enterprises. The first one has an administrative activity. It covers all the tunisian territory and has more than two hundred sites connected to the central site which is located in the capital. The remote connections between the central servers and the other sites require a high protection. The management of two security projects within this organization have been used in the initialization of the SECOMO model. The second enterprize provides security services to all the tunisian organizations. More precisely, it offers to them the service of securing sensitive data transfers. Confidentiality, authentication and non-repudiation are the main services offered by this agency. The management of a physical security project of this agency has been used in defining the SECOMO *a priori* model. Notice that these three projects have been performed by telecommunication engineers having two to seven years of experience in conducting security projects.

To define the SECOMO *a priori* model, we have used a regression tool referred to as UltimaCalc version 2.1.413. Input values provided to this tool are the $\ln(S)$ and $\ln(E)$ where S is the size of the security policy and E is the effort required to implement this policy. As output, UltimaCalc provides two parameters that are the slope and the intercept of the regression line. For the effort equation, the slope defines the $\beta + \sum W_i$ and the intercept is used to extract the $a \times \prod_i EM_i$ parameter. For the duration equation, the slope defines the $\delta + \sum W_i$ and the intercept is used to extract the c parameter. Table I shows the values of the parameters defining the SECOMO *a priori* model. Notice that the correlation coefficient for the effort and duration is respectively equal to 0.996195 and 1.

The effort and duration equations of the SECOMO *a priori* model are given by:

$$E \simeq 508.17 \times S^{0.07} \quad (6)$$

$$D \simeq 1.20e^{-9} \times E^{3.95} \quad (7)$$

B. Case study

The SECOMO *a priori* model has been used to manage a security project within the first organization described in the previous subsection. The project consists in installing a new firewall, securing the network traffic between the central

servers and the sites distributed in Tunisia, and testing the implemented solution.

To estimate the required effort and duration, we have started by describing algebraically the SP. Then, we have calculated the functionality which is shown in the *operation* and *predicate* parts of the algebraic specification. It was equal to $f = 100$. We have also calculated the complexity based on the *axiom* part of the formal policy. Its value was $c = 306$. The policy size is then $S = 321.925457$. According to the *a priori* model, the estimated effort is $E_{estimated} = 793.649316$ ManMinute and the estimated duration is $D_{estimated} = 352.871322$ minutes which is equivalent to 5.881 hours. The required number of persons as estimated by our model is then $P_{estimated} = 2.249$ persons.

The security project has been managed by the organization and the real values of the effort, duration and participating engineers were respectively $E_{real} = 720$ ManMinute, $D_{real} = 6$ hours and $P_{real} = 2$ engineers. Comparing the real values to the estimated ones, we notice that the SECOMO *a priori* model delivers close values compared to the real effort, the real duration and the real number of engineers assigned to the project. Table II gives the relative error between the estimated values and the real ones for this case study. However, since the estimations are made only using the *a priori* model, the estimation results will be different if performed after the management of several projects. We believe that these results will not vary a lot, if performed after the system has stabilized.

C. The SECOMO *a posteriori* model

The SECOMO model is adjusted each time a new security project is managed. The following steps are followed to adjust the model:

- Data gathering related to new security project management is performed. This activity is continuously realized in order to adjust the model.
- The *a priori* model and the statistical data are combined in order to define the *a posteriori* model.
- Security data gathering is continuously performed in order to adjust periodically the *a posteriori* model using the previous *a posteriori* model and the data newly collected.

In this paper, we have used the data relative to the security project management described in the case study to define the *a posteriori* model. The statistics have been added to the regression tool and we have obtained the values of the

TABLE I
A PRIORI MODEL DEFINITION

SECOMO parameter	Attributed value
$\beta + \sum W_i$	0.077208
$a \times \prod_i EM_i$	508.170996
$\delta + \sum W_i$	3.954301
c	$1.206735e^{-9}$

TABLE II
RELATIVE ERROR BETWEEN SECOMO *a priori* ESTIMATIONS AND THE REAL VALUES

	Estimated value	Real value	Relative error
Effort	793.649316	720	0.102
Duration	352.871322min	360min	0.019
Number of persons	2.249	2	0.124

parameters relative to the SECOMO *a posteriori* model. These values are given by Table III.

The effort and duration equations relative to the new SECOMO *a posteriori* model are given here after:

$$E \simeq 468.59 \times S^{0.08} \quad (8)$$

$$D \simeq 5.35e^{-8} \times E^{3.40} \quad (9)$$

D. The SECOMO *a posteriori* model adjustment

In addition to the previous example, we have considered an other one relative to a security project which has been performed by an internet provider. This agency is also responsible for the management of some domain of names. It also hosts some tunisian organizations web sites. The security project managed in this agency consists in securing the internal servers from unauthorized access coming from internal or external connections. We have formally represented the SP relative to this project. The functionality parameter of this formal SP is $f = 89$, and its complexity is $c = 228484$. Then the calculated size is $S = 486.214973$. Applying the SECOMO *a posteriori* equations, the estimated effort is $E_{estimated} = 793.030842$ ManMinute and the estimated duration is $D_{estimated} = 408.774391$ minutes or 6.81 hours. The estimated number of persons required to perform this project is $P_{estimated} = 1.940020$.

After managing the project, we have compared it to the real values. The latter indicate that the project has been performed by two engineers during six hours and a half that is 390 minutes. The real effort is then $E_{real} = 780$ ManMinute. The statistics relative to this project have been added to the regression tool in order to adjust the SECOMO *a posteriori* model. These values are given by Table IV.

The effort and duration equations relative to the new SECOMO *a posteriori* model are given here after:

$$E \simeq 464.60 \times S^{0.08} \quad (10)$$

$$D \simeq 5.58e^{-8} \times E^{3.40} \quad (11)$$

Table V gives the evolution of SECOMO parameters. We notice that the parameter values converge, somehow, as new

projects are considered. In fact, the increment added by each project is reducing. However, convergence will not be achieved with a low number of projects.

V. CONCLUSION

In this paper, we have addressed security project management. We have enhanced the SECOMO model by developing a new methodology for estimating security effort based on the algebraic specification of security policies and using the VSM technique. We have then defined the SECOMO *a priori* model and we have proved the accuracy of the new technique by applying it to a few security projects. We have also defined the SECOMO *a posteriori* model and we have adjusted it based on new security project management statistics. The integration of a large number of real statistics is mandatory to give a complete validation of the new methodology.

REFERENCES

- [1] J. Krichene, N. Boudriga, and S. G. El Fatmi, "SECOMO: An Estimation Cost Model for Risk Management", Proceedings of the 7th International Conference on Telecommunication (ConTel'03), pp. 593-599, Zagreb, Croatia, June 11-13, 2003.
- [2] B. W. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, "Software Cost Estimation with COCOMO II", Prentice Hall, 2000.
- [3] B. Boehm, R. Valerdi, J. A. Lane, A. W. Brown, "COCOMO Suite Methodology and Evolution", CROSSTALK The Journal of Defense Software Engineering, pp. 20-25, 2005.
- [4] R. Jeffery and G. Low, "Function Points and Their Use", Australian Computer Journal, vol. 29, no. 4, pp. 148-156, 1997.
- [5] T. E. Hastings, A.S.M Sajeev, "A Vector-Based Approach to Software Size Measurement and Effort Estimation", Proceedings of the IEEE Transaction on Software Engineering, vol. 27, no. 4, pp. 337-350, 2001.
- [6] S. Barman, "Writing Information Security Policies", New Riders, 2002.
- [7] J. Loeckx, H. D. Enrich and M. Wolf, "Specification of Abstract Data Types," Wiley Teubner, 1996.
- [8] M. Hamdi and N. Boudriga, "Algebraic Specification of Network Risk Management", Proceedings of the ACM Workshop on Formal Methods in Security Engineering, pp. 52-60, Washington, D.C., 2003.
- [9] M. Hamdi, J. Krichene and N. Boudriga. Algebraic Test Case Generation of Security Policies in Communication Networks, *Nordic Workshop on Secure IT Systems*, Tartu, Estonia., November 2005.
- [10] N. E. Fenton and M. Neil, "Software Metrics: Roadmap", Proceedings of the 22nd International Conference on Software Engineering , pp. 357-370, 2000.
- [11] N. E. Fenton and M. Neil, "Software Metrics and Risk", Proceedings of the 2nd European Software Measurement Conference (FESMA'99), pp. 39-55, Amsterdam, 1999.
- [12] G. Poels, "Towards a Size Measurement Framework for Object-Oriented Specifications", Proceedings of the 1st European Software Measurement Conference (FESMA'98), pp. 379-388, Antwerp, May 6-8, 1998.

TABLE III
A POSTERIORI MODEL DEFINITION

SECOMO parameter	Attributed value
$\beta + \sum W_i$	0.085042
$a \times \prod_i EM_i$	468.594108
$\delta + \sum W_i$	3.408602
c	$5.357445e^{-8}$

TABLE IV
A POSTERIORI MODEL ADJUSTMENT

SECOMO parameter	Attributed value
$\beta + \sum W_i$	0.085781
$a \times \prod_i EM_i$	464.602599
$\delta + \sum W_i$	3.402668
c	$5.587680e^{-8}$

TABLE V
SECOMO MODEL EVOLUTION

Nb	$\beta + \sum W_i$	$a \times \prod_i EM_i$	$\delta + \sum W_i$	c
3	0.077208	508.170996	3.954301	$1.206735e^{-9}$
4	0.085042	468.594108	3.408602	$5.357445e^{-8}$
5	0.085781	464.602599	3.402668	$5.587680e^{-8}$