Harmonizer – A Tool for Processing Information Security Requirements in Organizations

Jussipekka Leiwo, Chandana Gamage and Yuliang Zheng

Peninsula School of Computing and Information Technology
Monash University
McMahons Road, Frankston, Vic 3199, AUSTRALIA
Phone +61-(0)3-9904 4287, Fax +61-(0)3-9904 4124
E-mail: {skylark,chandag,yuliang}@fcit.monash.edu.au

Abstract. Information Security Requirement Harmonizer (shortly, harmonizer) is a software tool that aids information security personnel in organizations to formulate, organize and automatically harmonize various information security requirements from various sources within the development of information security. Various levels of abstraction are represented as the information security development organization is specified, and various types of information security requirements are associated to various components of the organization and transformed into the next level of abstraction in a manner which aims at reducing internal inconsistencies and therefore improving cost-efficiency and ease of the management of information security. This paper discusses theoretical foundation of harmonizer, as well as implementation and application details of the tool. The software is currently working and is publicly available.

1 Introduction

Information security requirements need to be studied at a number of levels of abstraction. Organizational information security objectives represent the highest level of abstraction of security policies, statements describing the general principles of how resources should be protected against various threats. Information security objective is a statement by top management that states the general objectives of information security in that specific organization. Highly abstract policies and objectives are refined and enforced by various levels of more concrete administrative, managerial, operational and technical information security policies that in concert form the backbone of information security in organizations. Information security policies describe the requirements that must be satisfied at each organizational level to fulfill the generic information security objectives [1].

It is clear, that lowest levels of security policies must be formal, for example as specified in various access control models. Therefore, the essence of layered security policies is the question of how to transform highly abstract organizational information security objectives into formally expressed security policies. The harmonizer approaches this problem by providing a mechanism to organize security policies according to the organization, and to automatically derive lower level policies from various partially specified information security requirements and harmonization functions, that are statements governing the reduction of abstraction.

Other approaches towards computerized support for the management of information security include, for example, decision support systems to aid in the specification of protection measures, as seen by the organizational information security chain [7]. The core of the approach is to identify the organizational information security chain and to capture the knowledge of information security to derive a check-list of required protection measures. As risk analysis has been the cornerstone of the management of information security, various approaches have been proposed to formalize the risk evaluation approach and provide software tools to support the analysis [2]. As full risk analysis is a costly process with uncertain results, some researchers have suggested that extensive check-lists would be a more suitable approach towards information security in most organizations [15]. Therefore, the above is well justified. Anyhow, check-lists have the following disadvantages:

1. Inflexibility in adopting into different organizational needs. Since lists are fixed and usually filled by marking those criteria that are taken into consideration, the process does not provide adequate support for application of same protection measures with different parameters.
2. Lack of support for assurance of internal consistency occurs when there is no means to formally optimize the requirement base to minimize the set of different measures applied, and to assure that different measures used are not in conflict with each other.
3. Little space for scientific research. Typically, measures to provide assurance from the security of systems have evolved from check lists through mechanistic engineering methods to formal modeling [4]. As returning from risk analysis to check-lists is actually a step backwards, not a step towards formal modeling, there is a need to develop approaches that could possible lead to modeling of security. Harmonizer approach attempts to provide abstractions of concepts modeled hence paving the way to formal modeling of information security and advancing the scientific knowledge of the subject area.

Therefore, a different approach is required. The rest of this paper is dedicated on the harmonizing approach. The next section first discusses the theoretical foundation of the approach. The software architecture for the harmonizer tool is the discussed in section 3. Issues related to the use of harmonizer shall be discussed in section 4. Conclusions shall be drawn and directions highlighted for future work in section 5.

2 Theoretical background

A formal specification for the model the harmonizer software is based on is given in [13] and [14]. Further properties are derived in [12]. Therefore, the focus shall not be on the re-specification of the formal model but on the discussion on the major characteristics from the software development and application point of view. After discussing the motivation for the modeling approach in section 2.1, the model shall be discussed in two parts: The security development organization in section 2.2 and specification and refinement of information security requirements in that organization on section 2.3.

2.1 Justification of modeling approach

From the early access control models, an interesting research question has been on the derivation of secure systems from security specifications. In 1989 it was shown by Jacob [9], as had for long being assumed to be the case, to be practically infeasible. Another related problem has been addressing the issue of composability of secure systems from secure components but it has proven to be feasible only with a limited set of security properties. Anyhow, as derivation of implementation and composability only deal with a limited subset of access control models, their applicability to the specification of comprehensive measures for information security in organizations is limited. Both access control and cryptographic policies can be expressed in various levels of abstraction, but most notations provide only little support for organizational dealing with information security requirements.

Various information system modeling tools, such as Z-notation for requirements engineering has been successfully applied in the specification of security properties of computer systems [5, 6], and other methods circulate among expressing duties and responsibilities of users, approaches ranging from the theory of normative positions [10] and responsibility modeling [8] to the view of information security as structures of responsibilities among systems [3]. All these models, anyhow, fail to meet the layered nature of security policies and provide only little support for multiple levels of abstraction of security policies.

A step towards modeling of information security has been taken by [2] where a methodology has been developed to formulate the organizational knowledge of the desired state of security (expressed as a policy) and the current state of information security and to bridge the potential gap or to prepare to the future changes in either policy or implementation. That model, anyhow, is very strongly dependent on risk analysis. As has been discussed earlier, risk analysis has recently been criticized quite heavily, and therefore the objective of this research has been to develop a model that is not too dependent on the justification of security, whether it being check-lists, risk analysis or some future development in the area. The focus shall be on the capturing of the knowledge concerning information security development organization, the desired protection and the rules of refining the statements concerning desired protection at various business units in the organization without loosing the guiding nature of upper level policies. The rest of this section describes how this is achieved.

2.2 Information security development organization

Information security policies are usually hierarchical to represent typical organizational structures. High levels of abstraction in the harmonizer model represent managerial layers in organizations and low levels of abstraction those where actual implementation and monitoring of information security measures are carried out. Therefore, the components of an organization, called business units or, shortly, units, are organized in hierarchical layers. An organization consists of a finite number of layers, and each layer of zero or more units. Each unit must belong to exactly one layer. Units are connected with *Child*-relationships, that are abstractions of a way requirements of a unit form the basis for requirements at a lower layer. Once there is a relation $(u_n^a, u_{n+1}^b) \in Child$ we say that there is a requirement mapping from unit u_n^a to unit u_{n+1}^b where unit u_n^a belongs to layer n and unit u_{n+1}^b to layer $n+1$. Each child of a unit belonging to a layer n must belong to layer $n+1$ to make sure no circular dependencies can occur.

Other components of the model are three types of requirements: harmonized requirements, unit-specific requirements and layer-specific requirements. Harmonized requirements are those that a unit passes to all of it's *Child* units for being refined. They are constituted from the harmonized requirements of that unit, unit-specific requirements that are applied only to that unit, and layer-specific requirements, that are applied to each unit belonging to a specific organizational layer. harmonized requirements have the highest priority since they represent the coordinating policy from upper layers. Next priority is with layer-specific requirements and the lowest with unit-specific requirements. These priorities are essential in deriving harmonized requirements as shall be discussed in the next subsection.

2.3 Specification and harmonization of requirements

Information security requirements within the harmonizer are expressed as statements of form

$$A; P; Q; Protocol; Algorithm; Parameters$$

where A is an association connecting processes P and Q. Process is an abstraction of any organizational element, whether it represents a process inside a computer, a computer, an information system, a business unit or entire division or other organization. Association A then represents a communication channel between the processes. It is assumed that communication is carried out According to *Protocol* that is a set of rules regarding formatting and sequencing of messages passed over association A. Protocol messages are then required to be protected by a security enforcement algorithm *Algorithm* as governed by a parameter vector *Parameters.* Parameter vector is of form

$$param = value; param = value; ...$$

where parameters that *Algorithm* requires are specified. The idea is to keep the number of different protocols and algorithms as minimum and coordinate the level of security enforced over different associations by parameters to reduce the number of security enforcement mechanisms that need to be maintained. This leads to an improved cost efficiency of protection.

Requirements are partially or fully specified statements of above form associated to that specific unit or layer. The requirements can be harmonized by two means: by merging them or by enforcing harmonization functions.

Merging of requirements is enforcement of layer specific and upper layer requirements into requirements of other units under the merging scope. When requirements are merged, either all units in a specific layer are altered according to layer specific requirements, or a unit enforces it's requirements into all of it's *Child* requirements in case they are in conflict. Types of conflicts of requirements are as follows:

1. *Association* and *Protocol* field of requirements match, but *Algorithm* is different. This conflict is simply removed by altering the *Algorithm* field of the lower priority requirement to be that of the higher priority requirement.
2. Any field of a requirement *Parameters* is different from a corresponding field of another requirement's *Parameters*. This conflict is removed by altering the lower priority requirement's value of that field according to that of the higher priority requirement.
3. Any parameter in a requirement is non-existent in another requirement. This conflict is removed by adding the parameter when associated to the corresponding value into lower priority requirement according to the *Parameters* vector of the higher priority requirement.

Once requirements are merged, they can be harmonized. Harmonization is a process of setting criteria that the requirement base must satisfy to meet the desired state, and enforcement of these conditions. Harmonization functions are specified and enforced in a specific scope in the organization. They can be applied to a specific unit, all units in a specific layer, a specific units and all it's descendants recursively, or to all units in the organization. Harmonization functions are specified as expressions

$$PreCondition : PostCondition$$

where each requirement in the given scope where *PreCondition* is true is altered according to *PostCondition*. More formal treatment of harmonization functions and their enforcement is given in [13]. *PreCondition* is one or more statements of form

$$[\ NOT\ ]\ field\ op\ value$$

connected with logical operators *AND* and *OR*. *field* can be any component of a requirement, an association name, process name, protocol name, algorithm name, or a name of a parameter. Depending on the field, *op* can be either equivalence symbol (=), greater-than symbol (>) or less-than symbol (<). *value* is any value that the value of *field* in each requirement on the harmonization scope. If the *field* holds numeric information all three operators are acceptable, in case of textual information, only equivalence is allowed.

PostCondition has three possible forms: *A field = value* is the basic form of requirement altering harmonization. The component of a requirement with a name *field* is altered to be *value*. For example, harmonization function

$$Algorithm = PGP\ AND\ kl < 1024 : A\ kl = 1024$$

can be used to inspect each requirement in the harmonization scope, and if that requirement states that *Algorithm* is PGP but the parameter *kl* is less than 1024, then all those *kl* fields are altered to be exactly 1024. This example makes sense if parameter *kl* is considered to be, for example, length of keys in PGP. Requirement altering harmonization can also be used to map requirements into lower level of abstraction, for example by giving various fields first generic names, such as *PK-Encryption* before specifying the actual encryption algorithm used, and later harmonizing away these generic names by replacing them with actual algorithm names and adequate parameters.

The second form of harmonization is requirement removing harmonization. In this case, the *PostCondition* is simply of form *R*. For example, harmonization function

$$Protocol = SMTP\ AND\ NOT\ Algorithm = PGP : R$$

simply removes each requirement where communication protocol is SMTP (E-mail) but requirement algorithm is not PGP.

Third is requirement generating harmonization that is used to deal with dependencies of requirements. In these harmonization functions, *PostCondition* is of form *C Req* where a new requirement *Req* is created and attached to the organizational component where the requirement belongs to. For example, the harmonization function

$$Algorithm = RSA : C\ A;P;Q;Proto;DSS;kl = 2048$$

can be used to identify each requirement where encryption algorithm is RSA and attaching a corresponding digital signature requirement into those requirements.

3 Software architecture

A software was developed using Java 1.1 to implement the theories described in the previous sections. A stand-alone version and an applet are publicly available at

```
http://pscit-www.fcit.monash.edu.au/~skylark/harm/
```

for review. This page also has links for technical documentation describing implementation details and user manuals giving detailed instructions on how to use the software.

The high level use case is illustrated in figure 1. The idea is, that security consultant formulates the organization and various requirements as directed by top management, and then enforces harmonization functions in that organization over the requirements. Anyhow, the knowledge of top management is not expected to be formulated by them directly. Managerial personnel is not expected to be familiar with details of formal presentations, even though the notation is intuitive, and their job is rather to state the target and objective of protection and pass it for various information security specialists for enforcement. Therefore, the informal knowledge of information security objectives is outside of the system scope. Therefore, information security requirement can be seen as *any formally expressed information security feature that a system must* *satisfy* to highlight the difference between informal information security objectives. Information security objectives become information security requirements once expressed using the notation described in previous sections. The use case scenario represents a harmonization cycle, a process that produces output from the various specifications for further consideration by security analyst. Further processing of the information, either by further harmonization or implementing desired protection measures, is then subject to the analyst's consideration. The intuitive knowledge regarding adequate state of a requirement base is not modeled.

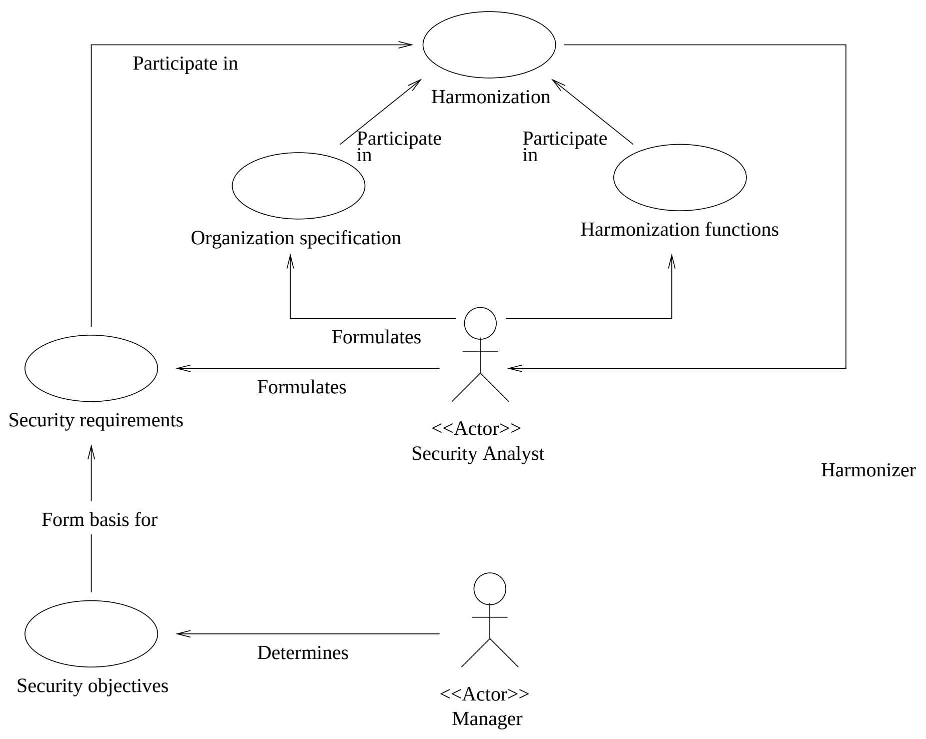

Fig. 1. A high level use case scenario for harmonizer

The high level class diagram is illustrated in figure 2. It should be noted that the actual implementation diagram is much more complicated but implementation details are available in technical documentation at the above URL. Also, some shortcuts have been taken in design as the current version is more a proof of concept than a commercial final product. *Precond* and *PostCond* in **harmFunction** and components of **Requirement** are merely represented as strings (*Parameters* of **Requirement** as a vector of strings) instead of separate classes. This simplifies design, even though it adds complexity of methods related to harmonization. Anyhow, these changes are quite trivial and do not reduce the expressiveness of harmonizer. Harmonizer software also includes a mechanism to document organizational security levels and to specify how various requirements can be interpreted within these levels. That is, anyhow, only for informative purposes and the levels are not formally enforced or used in harmonization, the structure is not included in the class diagram.

As the security development organization is strongly hierarchical, mapping it to a class structure is considerably straightforward. The main class is **Organization** that is an aggregate of classes **Layer** and **harmFunction**. Class **Layer** is further an aggregate of classes **Unit** and **LayerSpecificReq** that is a specialization of class **Requirement**. Class **Unit** is an aggregate of **UnitSpecificReq** that is a specialization of **Requirement**.

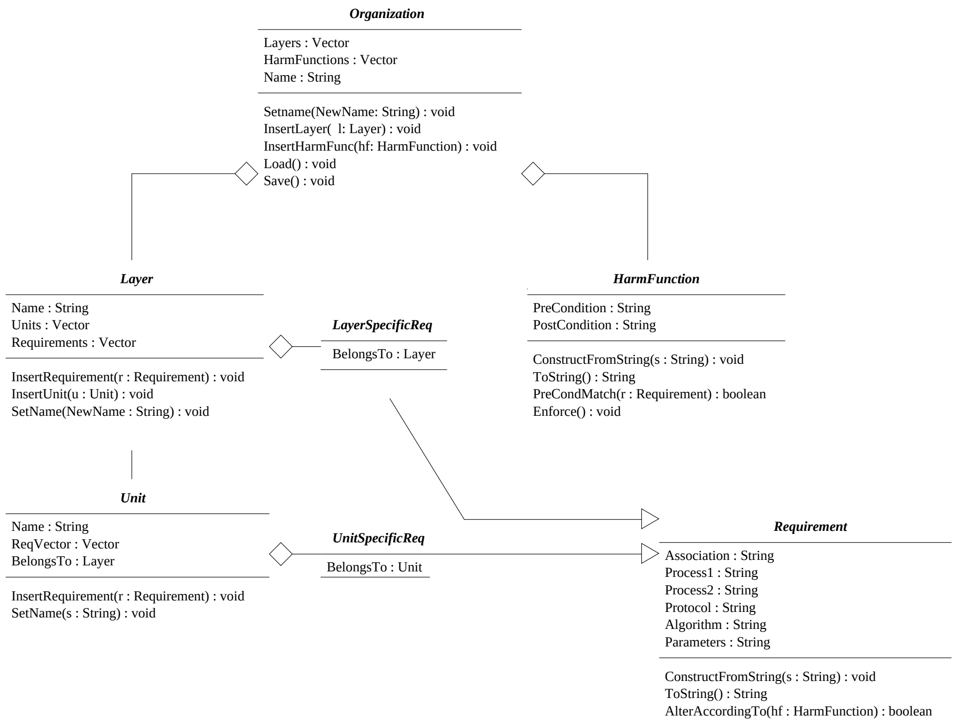

Fig. 2. High level class diagram of harmonizer

As full documentation for design, implementation and use of the software is available on-line, there is no need to study it in more detail herein. That documentation also justifies used language, summarizes problems identified and discusses various implementation alternatives of various functions. Instead, the important question of application of the software shall be addressed in the next section.

4 Application scenarios

The basic case of application is where the organization is first specified by each layer (identified by name), then adding units to each layer and specifying the relationships between layers, that is specifying *Child* relationships. Assume, for example, a simple three-layer organization illustrated in figure 3. Each layer is named to represent the type of layers they represent. To improve intuitiveness of the example, layers are named as *Top*, *Mid* and *Low* and each unit just numbered with the first letter of layer name indicating the layer they belong to. Arrows represent *Child* relationships in the organization.

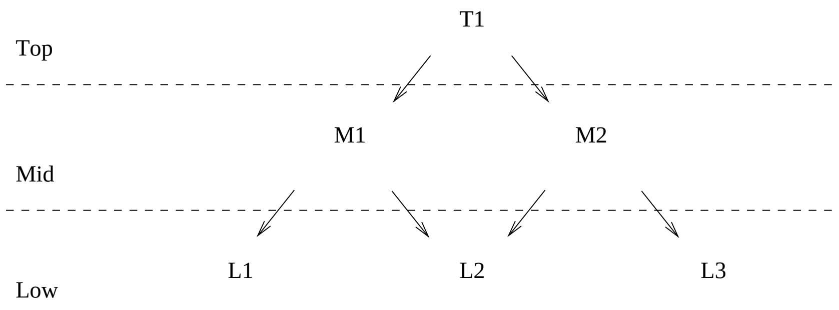

Fig. 3. An example of a three-layer organization

The next step is to specify layer specific requirements for the organization. To keep the example intuitive and simple, the encryption of Email is illustrated. At *Top* layer, there is only one unit *T1* to illustrate the top managerial layer in the organization. Let there be an informal information security objective *All internal Email communication must be protected by suitable encryption mechanisms against unauthorized eavesdropping.* The objective states roughly what is to be protected (internal Email communication), how (by suitable encryption mechanisms), and against which threats. Anyhow, there is a large amount of space for interpretation of what exactly is meant by statements like *suitable encryption mechanisms* and *unauthorized eavesdropping.* As the objective is stated by top management and is rather descriptive than specific (even if assisted by security specialists) it is the duty of security specialists to apply their knowledge to identify what exactly is meant by those statements.

Therefore, at the top layer, there may be a layer specific requirement expressed as

$$A;P;Q;Email;Encr;kl = unspecified$$

that gives generic names for all components of a requirement, This requirement can then be merged to all units of that layer (only *T1*) and in this case, that requirement becomes the only requirement of *T1*. Requirements of *T1* can then be merged to those of *M1* and *M2*. Let us assume there are no previous requirements, so the above requirement is directly copied into both *Mid* layer requirements. Anyhow, at this stage there may be more detailed knowledge of the type of communication technology used for specific communication paths. Assume, for example, that all E-mail communication between units *M1* and *M2* takes place over the Internet. There are two ways of enforcing these changes into requirement bases of *M1* and *M2*. Either by setting them as layer specific requirements for *Mid* layer or by specifying harmonization functions with the harmonization scope this layer only and enforcing them. As the latter is more intuitive, even though less structured, it shall be used. As each harmonization function only does an atomic harmonization task (alter, remove, or create), a number of them is required to implement the required changes.

First, it is known that the communication media is the Internet. That also means, that Email protocol is SMTP. Therefore, the two harmonization functions can be enforced throughout this layer. First one expressed as

$$Association = A : A\ Association = Internet$$

alters the association to be Internet, and the other one

$$Protocol = Email : A\ Protocol = SMTP$$

sets the communication protocol used. Next, two unit specific harmonization functions are executed for both *M1* and *M2*. For *M1* they are first

$$Process1 = P : A\ Process1 = M1$$

and

$$Process2 = Q : A\ Process2 = M2$$

and for process *M2* the same, except process names are swapped. Now the requirement base of *M1* looks like

$$Internet;\ M1;\ M2;\ SMTP;Encr;kl = unspecified$$

that is of reduced level of abstraction of that of original requirements at layer *Top*. Now, the next step is to specify what is the type of communication required for Email over the Internet. An obvious requirement is that it must be a public key method, let us say PGP. Now, more layer specific harmonization functions can be specified to deal with requirements. First, harmonization function

$$Association = Internet\ AND\ Protocol = SMTP : A\ Algorithm = PGP$$

sets PGP as the protection algorithm. Further, let there be a requirement that in case of PGP being used, minimum acceptable key length (expressed as parameter *kl*) is 1024 bits. This can be enforced by a harmonization function:

$$Algorithm = PGP\ AND\ kl < 1024\ OR\ kl = unspecified : A\ kl = 1024$$

that goes through the requirement base and if key length is unspecified or too short fixes it to 1024 bits. Note, that if there were a requirement (for example, as enforced in either of the units locally) that sets key length to be more than 1024, it is left unaltered. In some cases, it may be necessary to also restrict too large parameters to save time or processing power, or just to increase consistency throughout security enforcement, and exactly one acceptable key length, for example, is specified. To do this, a harmonization function can be specified as

$$Algorithm = PGP\ AND\ NOT\ kl = 1024 : kl = 1024$$
that alters each kl that is not previously 1024 to be 1024. Similarly to this, various optimizations and reductions of abstraction can be carried out at different layers but merging requirements with other requirements and then specifying harmonization functions to tune them up. At the end of the process, also organization wide harmonization functions can be executed to make sure there are no inconsistencies left in any of the requirements to assure that all key issues are taken into consideration.

As a result, each unit has it's own specific set of requirements that must be enforced at that layer. At upper layer the requirements are enforced directly by setting them as primitives for lower layers and then enforcing them at those units. As a result, each unit at technical layer should have a set of specifications using which security enforcement measures can be implemented and the process provides assurance from the state that all those measures are on align with organizational objectives and all coordination is centralized through the use of harmonization functions and requirement merging.

5 Discussion, conclusions and further work

The model and tool discussed in this paper contribute to the seamless derivation of technical security specifications from organizational information security objectives using the organizational knowledge for mapping between levels of abstraction. Therefore, the content and format of those objectives are out of scope, as are internals of security enforcement technologies. The content of security objectives can originate from check-lists, risk analysis or other managerial tools and are independent from the work reported herein. Also, academic and industrial effort in security enforcement technologies, especially formal access control models and cryptographic techniques has produced numerous well understood technologies to be used for secure system construction. Therefore, it can be assumed that standard technologies can be used for the actual security enforcement.

As security always increases cost of systems and operations, the essential research question addressed by the approach is on the optimization of security enforcement according to organizational standards. Automation in harmonization and merging of requirements together with the improved assurance on requirements that enforcement is based on is expected to reduce cost of secure operations. Further, it is of academic interest to take steps towards formal modeling of information security, and some first steps towards formal models for organizational information security are taken by providing means to capture the information security know-how and formally deriving information security requirements using that knowledge.

There are two major drawbacks in the model: First is the dependency on the organizational hierarchy and second is the lack of intuitive modeling tools. As the emergence of electronic commerce and new types of flexible organizational structures reduces the hierarchy traditionally seen as the requirement for effective information security, the model may in the future need to be adjusted into more flexible specification of organizations. On the other hand, automated treatment of requirements and harmonization functions would assist in rapid adaptation into changing processing environment by means of rapid alteration of security specifications. Further, providing an organization free presentation of common sets of typical security requirements might also increase the speed and reduce cost of security maintenance in changing conditions. Due to the success of extensive check-lists and baseline protection criteria, it can be assumed that the core of security requirements are similar in many organizations, hence various sets of operational domain specific information security requirements might increase the cost-efficiency of information security.

Another open issue is in the development of a modeling tool to support dealing with the model. As current mechanisms to express organizations, requirements and harmonization functions are not very intuitive but depend on the understanding the way formal treatment of them is organized, there is a risk that the applicability of the theory suffers from this. Also, more intuitive modeling techniques would improve the communication between various parties involved in the process, as well as support integration of security design into general system design. Due to the hierarchical structure of the way various components of the model are organized, various object oriented modeling techniques might provide good solutions for this. Early steps towards this direction have been taken in [11] but the issue is still under active research.

References

1. M. D. Abrams and D. Bailey. Abstraction and refinement of layered security policy. In *Information Security. An Integrated Collection of Essays*, pages 160–170. IEEE Computer Society Press, 1995.
2. A. Anderson, D. Longley, and L. F. Kwok. Security modelling for organizations. In *2nd ACM Conference on Computer and Communications Security*, pages 241 – 250, Fairfax, VA, USA, 1994. ACM Press.
3. J. Backhouse and G. Dhillon. Structures of responsibility and security of information systems. *European Journal of Information Systems*, 5(1):2 – 9, 1996.
4. R. Baskerville. Information systems security design methods: Implications for information systems development. *ACM Computing Surveys*, 25(4):375–414, December 1993.
5. A. Boswell. Specification and validation of a security policy model. *IEEE Transactions on Software Engineering*, 21(2):63 – 68, February 1995.
6. E. Dubois and S. Wu. A framework for dealing with and specifying security requirements in information systems. In *Proceedings of the IFIP TC11 11th International Conference on Information Systems Security.* Chapmann & Hall, 1996.
7. T. Finne. A DSS for information security analysis: Computer support in a company's risk management. In *1996 IEEE International Conference on Systems, Man and Cybernetics vol. 1 of 4*, pages 193–198, Beijing, China, 1996.
8. R. Grimm. A model of security in open telecooperation. In *Proceedings of the IFIP TC6/WG6.5 International Conference on Upper Layer Protocols, Architectures and Applications*, IFIP Transactions C: Communication Systems, pages 425 – 440, Vancouver, B.C., Canada, 1992. North-Holland.

9. J. Jacob. On the derivation of secure components. In *1989 IEEE Symposium on Research in Security and Privacy*, pages 242 – 247. IEEE Computer Society Press, 1989.
10. A. J. I. Jones and M. Sergot. Formal specification of security requirements using the theory of normative positions. In *Computer Security – ESORICS92. Second European Symposium on Research in Computer Security*, number 648 in Lecture Notes in Computer Science, pages 103 – 121, Toulouse, France, 1992. Springer-Verlag.
11. J. Leiwo, C. Gamage, and Y. Zheng. An object–oriented modeling approach into the management of information security. In *Proceedings of the IFIP TC11 14th International Conference on Information Systems Security*, Vienna, Austria and Budapest, Hungary, September 1996. Chapman & Hall.
12. J. Leiwo, C. Gamage, and Y. Zheng. A mandatory access control policy model for information security requirements. In *Computer Science. Proceedings of the 21st Australasian Computer Science Conference, ACSC'98*, Australian Computer Science Communications Vol. 20 Nr. 1, pages 527 – 538, Perth, WA, Australia, 1998. Springer.
13. J. Leiwo and Y. Zheng. A formal model to aid documenting and harmonizing of information security requirements. In *Information Security in Research and Business. Proceedings of the IFIP TC11 13th International Conference on Information Security*, pages 25 – 38, Copenhagen, Denmark, 1997. Chapmann & Hall.
14. J. Leiwo and Y. Zheng. A framework for the management of information security. In *Information Security – Proceedings of the First International Workshop*, number 1396 in Lecture Notes in Computer Science, pages 232–245. Springer–Verlag, 1997.
15. R. vonSolms. Information security management: The second generation. *Computers & Security*, 15(4):281 – 288, 1996.