A Mandatory Access Control Policy Model for Information Security Requirements

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. Two basic paradigms towards specification of information security requirements can be taken: continuous specification and early specification of requirements. In models supporting continuous specification and refinement of information security requirements, the development organization is more vulnerable to the tampering with partially specified requirement primitives. This paper proposes a formal model for requirement access control to prevent unauthorized modification of security requirements, that may lead to weak or inconsistent implementation of protection measures.

1 Introduction

Models for the development and management of information security can be classified according to the phase of the specification and formulation of information security requirements. Many models support early specification of security requirements as a starting point for the development. Early risk analysis based models [Parker, 1981; Fisher, 1984] and some recent development frameworks such as [Booysen and Eloff, 1995] suggest that security requirements can be identified and formulated, and later phases of the work can be based on these requirements. These models are applicable when integrating security into existing systems or executing a periodical revision of information security. When developing systems where information security is an integral system objective, early specifications of security requirements may not be easily applicable or may lead to a very high level abstract description of security objectives not capable of clearly expressing the desired protection. Security requirements intent on tackling specific threats that originate from underlying technology, implementation technologies, system architectures and development tools that need to be decided upon before security requirements can be formulated. As high level descriptions at early phases of development of systems focus on technology-independent abstractions of systems, the assumption of early specification of security requirements is in conflict with the fundamentals of system design.

Models such as [Leiwo and Zheng, 1997a] have been developed to take into account the multiple sources of security requirements and the need for continuous revision of security requirements by reduction of abstraction from incomplete high level security objectives until concrete protection measure specifications are reached. The major focus of the development of information security is on the processing of requirements, and the exact point of the transformation from requirements into protection specifications becomes unclear, whereas in early specification approaches the focus is on the specification of countermeasures against potential threats by making clear the difference between a requirement and a protection measure. The continuous refinement approach leaves more flexibility in the specification of protection measures by allowing other essential characteristics of systems to be specified and making security requirements conditionally dependent on these characteristics.

Information security requirements should be classified very sensitive since they may disclose weaknesses of systems security [Brinkley and Schell, 1995]. Continuous processing of requirements makes the system vulnerable to unauthorized disclosure or modification that may lead to an intentional reduction of the level of system security, hence making it more vulnerable to deliberate attacks. To prevent unauthorized modification, an access control policy model shall be proposed for information security requirements. Access control policy model [Boswell, 1995] is an implementation-independent formalization of computer security requirements. The paper starts by a brief introduction of the background of this research in Sect. 2. The Major contribution of the paper is in Sect. 3 where a detailed model is proposed for preventing unauthorized access towards resources. The approach shall be evaluated, conclusion drawn and directions highlighted for future work in Sect. 4.

2 Background and motivation

2.1 Flow of requirements

This paper is based on the formal model for harmonizing information security requirements [Leiwo and Zheng, 1997a]. The model is mostly concerned with flows of information security requirements between organizational units and systems. Each unit belongs to a particular organizational layer and the major concern of the management of information security is the specification of harmonization functions that reduce abstraction and harmonize, optimize and resolve conflicts from security requirements originating from several sources. Several organizational layers are organized into a hierarchy, and each of these layers has a set of units dedicated to some specific task within the management and development of information security. Uppermost layers represent high level security objectives, dependent on the operating environment of the organization, and lower layers represent concrete implementation and operational duties regarding the security of systems. Based on different requirements at each layer, abstraction is reduced from security requirements until the lowest layers, concrete implementation specifications, are reached. At this stage, a harmonized, consistent and free of conflict set of protection requirements should be specified to each system and their interconnection mechanism.

An organizational security model for an N-layered security development organization can be presented as a 4-tuple (L, U, I, S), where $L = \{L_1, L_2, \ldots, L_N\}$ is the set of layers within the development organization, $U = \{u_{i,j} | i = 1, 2, \ldots N, j = 1, 2, \ldots Count(i)\}$ are units that get requirements and modify them for further refinement by units at lower layers. Function $Count(i)$ returns the number of units at layer L_i. $I = \{I_1, I_2, \ldots I_N\}$ within the model are the layer-specific requirements, and $S = \{s_{i,j} | i = 1, 2, \ldots, N, j = 1, 2, \ldots Count(i)\}$ are the unit-specific requirements. Harmonization refers to the specification of requirements R using vertical and horizontal harmonization functions, τ and ρ. Vertical harmonization refers to the modification of requirements so that very high level strategies are step by step refined to more concrete specifications of protection measures and horizontal harmonization refers to activities that identify similar requirements at each layer and guarantee interoperability of connected systems at each layer.

Two essential properties of the model are $Child$ and $Parent$ relations. We say, that there exists a relation $Parent(u_{i,j}, u_{i',j'})$, or $u_{i,j} = Parent(u_{i',j'})$ where exists a harmonization mapping $u_{i,j} \rightarrow u_{i',j'}$. Similarly, a relation $Child(u_{i,j}, u_{i',j'})$, or $u_{i',j'} = Child(u_{i,j})$ refers to the case, where exists a harmonization mapping $u_{i,j} \rightarrow u_{i',j'}$. Obviously, if each harmonization mapping reduces the abstraction of requirements, that means requirements are transferred to a lower level of abstraction, it holds that if $u_{i,j} \in L_i$, then $Child(u_{i,j}) \in L_{i+1}$ and $Parent(u_{i,j}) \in L_{i-1}$. Also, it is possible to have several $Parent$ and $Child$ units, leading to an increased need for harmonizing by potentially introducing conflicts between upper level requirements.

Let $R_{i,j} \in R$ be a requirement of unit $u_{i,j} \in U$ at layer $L_i \in L$, $I_i \in I$ be the layer specific requirements of layer L_i, and $S_{i,j} \in S$ be unit specific requirements of unit $u_{i,j}$. $\tau_{i,j} : R \times I \times S \rightarrow R$ are functions that implement the actual revisioning of requirements so that each $\tau_{i,j}$ is a vertical harmonization function to map requirements $R_{i,j}, I_i$, and $S_{i,j}$ to requirements $R_{i+1,j'}$ where each unit $u_{i+1,j'} \in Child(u_{i,j})$. Function $\tau_{i,j}$ is as $\tau_{i,j}(R_{i,j}, I_i, S_{i,j}) = R_{i+1,j'}$ for each $u_{i+1,j'} \in Child(u_{i,j})$. Vertical harmonization is specification of functions ρ_i on a given layer L_i. Highly conceptual harmonization functions can be implemented by first order logic statements specified in [Leiwo and Zheng, 1997b].

2.2 Security of requirement flows

The requirement flow between different layers in an organization is secure, when unauthorized entities can not violate confidentiality or integrity of information security requirements. This protection is essential, since in the development of systems with very high level security requirements, especially when requirement processing is automated or distributed, heuristic and informal methods of providing assurance of security specifications meeting organizational security objectives are not adequate. Instead, a formal proof is required. Several models and logics exist for establishing and testing the security of systems but the approach taken in this report is significantly different. Instead of protecting information flows within a computer system, the focus will be on the protection of the design of documents of a secure system. This protection is essential [Brinkley and Schell, 1995] but has not been formally approached.

Our threat scenario consists of unauthorized entities gaining access to security development documents, and then either disclosing those documents or modifying them to reduce the level of target security. Confidentiality of requirement flows is protected when unauthorized entities are not capable of disclosing sensitive security specifications, and integrity is protected when no unauthorized entity may modify requirements to reduce the level of security within the system. Typically computer security refers to the protection of confidentiality, integrity and availability. As has been shown by [Harrison et al., 1976], access control based prevention of denial of service, that is enforcement of availability, is undecidable and a feasible approach must be based on resource allocation. It is, anyhow, not within the scope of this paper to study denial of service in detail.

Assume, for example, a distributed database management system (DBMS) with security administration unit, where a central control is established to coordinate security administration units. This is illustrated in Fig. 1. Security administration may consist of several layers, each in charge of adapting central security requirements to the specific needs and operating at different levels of abstraction. Information security requirements flow between units within the security administration sub system, each layer operating at different level of abstraction. Assume a Trojan horse planted into the system, with an intention to modify security requirements in order to increase system vulnerability to an attack in future. A malicious requirement access is illustrated using dotted line, whereas solid lines represent authorized access. Corrupted unit, due to malicious access on requirements, has also been illustrated using dotted line. This corruption then leads indirectly to a reduced level of security in a database.

The higher in the organization the change occurs, the more the impact propagates throughout the organization. Therefore, the more there are chances for malicious units to violate security of requirements. Discretionary access control models are not capable of preventing sophisticated Trojan horse attacks, so a mandatory approach is needed. Assume that *Parent* and *Child* relationships are available from a centrally maintained database with a limited functionality and tamper proof implementation. This database is used to specify *Parent* and *Child* relationships, and the security of access into requirements is based on it. Therefore, assuming the malicious unit within the illustration attempts either to learn requirements in an unauthorized manner, or to modify requirements in an unauthorized manner, such behavior can be identified and prevented by a policy model proposed in this paper.

2.3 Modeling approach

Early access control models for confidentiality [Bell and La Padula, 1973] and integrity [Biba, 1977], as well as information flow models [Denning, 1976] have lead the way to a number of lattice based security models [Sandhu, 1993] and various models have been proposed to provide more flexibility on specifying authorizations [Jajodia et al., 1997]. Recent results support strength of role based

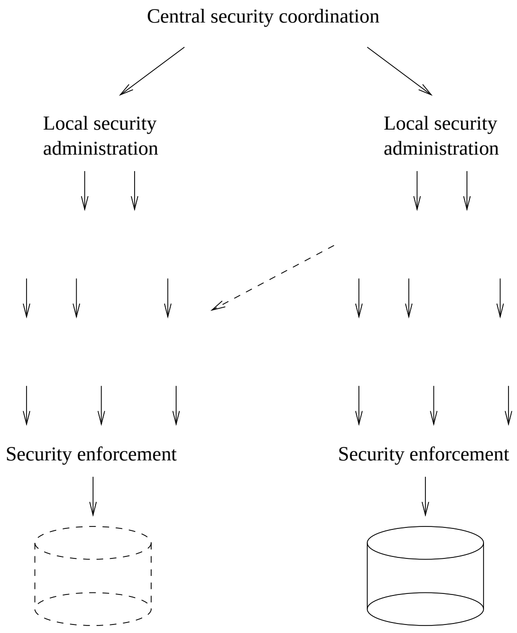
Central security coordination
Local security administration
Local security administration
Security enforcement
Security enforcement

Fig. 1. Threat scenario for a distributed DBMS

access control (RBAC) models [Sandhu et al., 1996] over traditional models. The fundamental difference is that instead of specifying rules governing authorized access between subjects and objects, organizational roles are analyzed, and constant authorizations are attached to each role. Users are assigned to different roles depending on the tasks they are executing. This intended to simplify the role of security administrators and to provide them with more flexibility in the specification of access rights.

The approach taken within this paper is to specify a role based mandatory access control policy model for the security development organization. To simplify the presentation, the notation for access control rules shall be given in the (s, o, r) form, that is intuitively interpreted as a subject s is granted r-access to an object o. Subjects represent active entities, usually acting on behalf of users and objects represent passive data objects or other resources to be accessed by subjects. Several types of access rights can be specified, such as *write, read, execute, append, remove* but we shall only be concerned with a minimum set of *read, write* and *append* accesses. *Execution* and *removal* rights are explicitly assumed to be authorized only to the owner unit. Each organizational unit acts as a role of $Parent$ or $Child$ regarding the type of access, and access rights to requirements are based on this role. Our model is mandatory, since units in roles are not authorized to independently specify or alter access rights of requirements. An alternative approach would be to specify a discretionary policy model, but this is a trivial task and need not be considered herein.

3 Protection of requirement flows

Let the set $H = \{h_1, h_2, \dots h_k\}$ be the *harmonization flow* within the N-layered development organization that consists of k *harmonization threads.* Each thread consists of $N-1$ *harmonization mappings.* A harmonization mapping (or, for simplicity, a mapping) is of form $u_{i,j} \rightarrow u_{i+1,j'}$ where it must be that $u_{i,j} \in Parent(u_{i+1,j'})$. This means, that a harmonization thread is of form $u_{1,j_1} \rightarrow u_{2,j_2} \rightarrow \cdots \rightarrow u_{N,j_N}$ where each $u_{k,j_k} \in Parent(u_{k+1,j_{k+1}})$.

Security of requirement flows refers to the non-existence of mappings that violate the security of any harmonization thread. Requirement integrity of a requirement flow is satisfied if write and append access to requirements of a unit are only granted to $Parent$ units of that unit. Integrity as a general concept means that information can only be modified or removed by an authorized entity [iso, 1988]. In the harmonized development model, it is assumed, that only $Parent$ units of a unit are authorized to write or append its requirements. Requirement confidentiality of a requirement flow is satisfied if read access to requirements of a unit is only granted to $Child$ units of that unit. Confidentiality as a general term refers to the property of information being accessible only upon an authorized request. Within the model, requirement confidentiality refers to the property of requirements, where only $Child$ units of a unit are authorized to access its requirements for reading. More exact definitions for requirement integrity and requirement confidentiality shall be given in Sect. 3.1.

Requirement integrity and confidentiality can be enforced by establishment and enforcement of a requirement access control policy, analyzed in Sect. 3.1. This is, anyhow, not enough for the development of systems with high security requirements. In addition to requirement integrity and confidentiality, requirement non-interference must be enforced. This shall be studied in Sect. 3.3. Basically, requirement non-interference means, that some units may need to enforce *requirement independence.* This refers to the prevention of *conflicting requirements*, specified in Sect. 3.2. Before formal analysis, intuitive definitions shall be given to requirement independence and requirement non-interference. We say, that a unit satisfies requirement independency if none of the units within the requirement scope of that unit are modified by the units that are outside of the requirement scope. Assume, that a unit within the model has high security requirements, and it considers only those requirements trusted that it has full control of. Basically, this means that no requirement outside of the scope of that unit may modify requirements within that scope. The *requirement Scope* of a unit refers to the collection of requirement threads from that unit onwards. When requirement non-interference is concerned, we say that a requirement flow satisfies requirement non-interference if none of the requirement mappings violates the requirement independency of requirement independent units.

3.1 Requirement integrity and confidentiality

Requirement integrity and confidentiality can be protected by establishing and enforcing a *requirement access control policy* for the requirement flow. Assume, that each unit $u_{i,j} \in U$ is acting both as a subject and an object for a *harmonization request* where a subject is accessing an object for exchanging information concerning harmonization. The notation shall be used, where $u^s_{i,j}$ refers to the unit $u_{i,j}$ being a subject of a request for a harmonization operation *op*, and $u^o_{i,j}$ refers to the unit $u_{i,j}$ being an object of a request for a request for harmonization operation. Subjects are considered as active entities accessing objects for an operation.

The access modes that are considered within the harmonization, are a subset of access modes within the Bell-LaPadula (BLP) model [Bell and La Padula, 1973]. Since the model for harmonized development of information security [Leiwo and Zheng, 1997a] is concerned with the exchange of information, the execution of an object is not applicable. Instead, three access modes *read,append*, and *write* are applicable. For integrity and confidentiality, the major concern of the BLP model, is the specification of the *access set b*, that is composed of triples $\langle u^s_{i,j}, u^o_{i',j'}, op\rangle$ that refers to the acceptable access modes *op* that subject $s^s_{i,j}$ is authorized to access object $u^o_{i',j'}$. To protect requirement confidentiality, condition 1 should be satisfied.

Condition 1 (Requirement Confidentiality) $\forall u^s_{i,j}, u^o_{i',j'} \in U : (u^o_{i',j'} \in Child(u^s_{i,j})) \Rightarrow (\forall\langle u^s_{i,j}, u^o_{i',j'}, op\rangle \in b : op \neq read)$

The significant difference between requirement confidentiality and confidentiality within the BLP model is, that in the BLP model, protection is based on security domains, but in the requirement confidentiality, on *Child* relation. So, requirement confidentiality establishes a NRC (No Read of Child) condition. To enforce confidentiality of requirements, that is prevent units from unauthorized disclosure of requirements, the only protection method is to prevent them from reading requirements from their *Child* units. Assume that there exists a unit that gets requirements from two separate *Parent* units. Now, to prevent *Parent* units from overriding requirements established by each other, they shouldn't be able to disclose each other's requirements by reading the requirements from the *Child* unit common to them. Requirement confidentiality prevents this type of violation.

Requirement access control policy for requirement integrity is influenced by the Biba model for integrity of information [Biba, 1977]. To protect integrity of information within the model for harmonized development of information security, NWP (No Write to Parent) protection shall be established. The requirement flow satisfies NWP requirement if condition 2 is satisfied. Again, prevention of read and append access modes is based on the *Parent* relationship, not according to the security classification. Consider, for example, a case where a unit attempts to falsifely override protection mechanisms required by one of its *Parent* units. Prevention of such modification is essential to enforce the security of features established by particular requirements.

Condition 2 (Requirement Integrity) $\forall u^s_{i,j}, u^o_{i',j'} \in U : (u^o_{i',j'} \in Parent(u^s_{i',j'})) \Rightarrow (\forall\langle u^s_{i,j}, u^o_{i',j'}, op\rangle \in b : ((op \neq write) \wedge (op \neq append)))$

We now say that a harmonization flow satisfies requirement confidentiality if condition 1 is satisfied and that a harmonization flow satisfies requirement integrity if condition 2 is satisfied. Enforcement of these conditions is not studied within this paper.

3.2 Conflicts of requirements

To study conflicting requirements, an exact specification needs to be given for requirement independence. Assume, that a unit $u_{i,j} \in U$ is declared to require requirement independence. Requirement independence is enforced, if no unit $u_{i',j'}$, where exists a harmonization thread $u_{i,j} \rightarrow u_{i+1,j_1} \rightarrow \ldots \rightarrow u_{i',j'}$, where units $U' = \{u_{i,j}, u_{i+1,j_1}, \ldots, u_{i',j'}\}$ are accessed, gets requirements from unit $u_{i'',j''} \notin U'$. To verify whether a unit $u_{i,j}$ enforces security independence, function Θ shall be specified in equation 1. Unit $u_{i,j}$ satisfies requirement independency if $\Theta(i,j) = True$.

$$\Theta(i,j) = \begin{cases} True, if\ (i = N-1) \wedge \forall u_{N,j'} \in Child(u_{i,j}) : \\ \qquad \forall u_{i'',j''} \in Parent(u_{N,j'}) : u_{i'',j''} \in U' \\ False, if\ \exists u_{i',j'} \in Parent(u_{i,j}) : u_{i',j'} \notin U' \\ \bigwedge_{u_{i',j'} \in Child(u_{i,j})} \Theta(i',j')\ Otherwise \end{cases} \tag{1}$$

Obviously, if unit $u_{i,j}$ does not satisfy requirement independency, there exists one or more conflicting requirements that enable the requirement independency of $u_{i,j}$. In addition to the conflicts in requirement independency, the conflict may be in the security level a particular requirement satisfies. Systems with multi-level security (MLS) need to enforce several security levels. As protection always increases the cost of processing, and the objective is to guarantee protection with as small costs as possible, functions and mechanisms operating on information with different classification needs to enforce different levels of security. Assume two security classifications, C_1 and C_2 where C_1 dominates C_2. Security of C_2 is not violated if processed with functions and mechanisms enforcing C_1 level security, but unnecessary costs will be caused, since functions and mechanisms enforcing C_2 but not C_1 would cost less and still provide adequate protection for C_2.

Other type of conflicting requirements, *conflict of security level* can be prevented by establishing a classification for each object $u^o_{i,j}$. This classification is also the clearance of $u^s_{i,j}$, when unit $u_{i,j}$ is acting as a subject. The significant property of classifications and clearances of units is, that in addition to providing a classification and clearance to the unit, they indicate the level of security of information that the mechanism or function they establish is capable of processing. If unit $u_{i,j}$ dominates unit $u_{i',j'}$ it has an intuitive meaning, that the mechanism that $u_{i,j}$ establishes is capable of processing information with higher classification than that established by $u_{i',j'}$.

To give an exact specification of a conflict in security level, function Φ shall be formulated as in equation 2. Unit $u_{i,j}$ is free of conflicts in security level if $\Phi(i,j) = True$. Boolean function $Dom : U \times U \rightarrow \{True, False\}$ shall be established to assist in the specification of Φ. $Dom(u_{i,j}, u_{i',j'}) = True$ if the classification of $u_{i,j}$ is equal to or higher than classification of $u_{i',j'}$ and $False$ otherwise.

$$\Phi(i,j) = \begin{cases} True \;\; if \; \forall u_{i',j'} \in Parent(u_{i,j}) : Dom(u_{i',j'}, u_{i,j}) \\ False \; if \; \exists u_{i',j'} \in Parent(u_{i,j}) : Dom(u_{i,j}.u_{i',j'}) \end{cases} \tag{2}$$

Intuitively, function Φ states that in any requirement mapping $u_{i,j} \rightarrow u_{i',j'}$ the classification of unit $u_{i',j'}$ may only be equal to or higher than the classification of $u_{i,j}$. In the sense of security enforcement, this does not cause any conflicts, but the cost-effectiveness of protection measure may reduce if some requirements establish higher level of protection than needed to satisfy system security objectives.

3.3 Protection of conflicting requirements

As defined in Sect. 3.2, two types of conflicts may occur: conflicts against requirement independency and conflicts against requirement classification. Let $\sigma_{i,j}$ refer to the classification of unit $u_{i,j}$. Let $C = \{c_1, c_2, \ldots c_n\}$ be the set of security levels of the development organization. C should be organized so that there exists a partial order $(C, \leq)$, where $c_1 \leq c_2 \leq \cdots \leq c_n$. Intuitively, this means that class c_k is provide with higher security classification than c_l, if $k < l$. Also, we say that $\sigma_{i,j} \in c_k$, if a requirement at unit $u_{i,j}$ is provided with security class c_k. To establish a protection against requirement conflicts, it must be that every unit belongs to exactly one security class $U_k \subseteq U$, so that formulae 3 and 4 are valid.

$$U = \bigcup_{k=1}^{n} U_k \tag{3}$$

$$\forall i,j : 1, 2, \ldots, N : (i \neq j) \Rightarrow (U_i \cup U_j = \emptyset) \tag{4}$$

Now we say, that $u_{i,j}$ dominates $u_{i',j'}$, if $u_{i,j} \in U_k$ and $u_{i',j'} \in U_l$, where $k \leq l$. This is same as $\sigma_{i,j} \leq \sigma_{i',j'}$.

Intuitively, prevention of conflicting requirements means that there shouldn't be any mapping $u_{i,j} \rightarrow u_{i',j'}$ that might violate requirement independency or where the classification of $u_{i,j}$ is lower than $u_{i',j'}$, that is $\sigma_{i',j'} < \sigma_{i,j}$. Scenarios of the result of such violations are numerous. Consider, for example, that requirement $R_{i,j}$ is provided with an encryption of highly sensitive information. Now, if $R_{i',j'}$ provides with encryption of less sensitive information, and there is a requirement flow from $u_{i',j'} \rightarrow u_{i,j}$, where a weaker property is established, than that required by $u_{i,j}$, the processing of higher level security may be seriously violated. To prevent conflicts of requirements, a requirement access control policy shall be establish to prevent requirement flow, that is write and append access, $u_{i,j} \rightarrow u_{i',j'}$ where $\sigma_{i',j'} < \sigma_{i,j}$. So, condition 3 should be satisfied. A harmonization flow satisfies prevention of classification conflicts of requirements if condition 3 is satisfied.

Condition 3 (Classification conflict of requirements)
$\forall \sigma_{i,j} < \sigma_{i',j'} : (\forall \langle u^s_{i,j}.u^o_{i',j'}, op\rangle \in b) \Rightarrow ((op \neq read) \wedge (op \neq append))$

Intuitively, requirements can only flow from lower classifications to higher. This may cause a reduction in cost effectiveness, but doesn't violate the security of the system. There are two ways to improve the cost effectiveness. Either, a requirement access control policy can be established to prevent such flows, or horizontal harmonization functions ρ (see Sect. 2) can be specified to act as a down grader, where unnecessary high classified requirements are modified to enforce only the level of security that is required by the target level. Preventing flows is analogous to condition 3. Instead of preventing harmonization mappings $u_{i,j} \to u_{i',j'}$ where $\sigma_{i',j'} < \sigma_{i,j}$ also such mappings should be prevented in every case $\sigma_{i,j} \neq \sigma_{i',j'}$. Specification of harmonization functions is the more effective method, but shall not be studied in detail within this report. The major philosophy within the horizontal harmonization is the reduction of complexity of the design by identifying similar requirements within a given layer, and providing a harmonized approach to these requirements.

To prevent conflicts of requirement independency, assume that there is a set $U'' \subseteq U$, where $U'' = \{u_{i_1,j_1}, u_{i_2,j_2}, \ldots, u_{i_n,j_n}\}$ is a set of all units requiring requirement independency. Based on U'', each subset $U''_k \subset U''$ can be established so, that each U''_k contains unit u_{i_k,j_k} and all units $u_{i',j'}$ where there exists a harmonization thread that accesses u_{i_k,j_k} and $u_{i',j'}$. To prevent conflicts of requirement independency, it should be that if l is the number of requirement independent units, then $\forall i,j : 1,2,\ldots,N : (i \neq j) \Rightarrow (U''_i \cap U''_j = \emptyset)$.

A requirement access control policy to support requirement independency should prevent any kind of access where subject $u^s_{i,j}$ is accessing object $u^o_{i',j'}$ in any access mode.A requirement flow satisfies requirement independency if condition 4 is satisfied.

Condition 4 (Requirement independency)
$\forall u^s_{i,j} \in U''_{k_1}, u^o_{i',j'} \in U''_{k_2}, k_1 \neq k_2 : \forall \langle s, o, op\rangle : (s \neq u^s_{i,j}) \wedge (o \neq u^o_{i',j'})$

The fundamental assumption here is, that access for subject s to object o in access mode op is granted if exists a triple $\langle s, o, op\rangle \in b$. If such a triplet doesn't exist, access should be denied by definition.

4 Evaluation, conclusions and future work

A model has been proposed for specifying and enforcing a mandatory requirement access control policy for the processing of information security requirements. The value of the proposal lies in three major issues: provision of a foundation for secure access to information security requirements, provision of evidence of applicability of the generic security framework and improving a scientific approach towards management of information security. First, the model establishes a formal foundation of secure automated processing of information security requirements in distributed organizations. Applications areas include multinational corporations and new types of organizations such as virtual and adaptive organizations, where computing paradigms are based on networking and frequent changes in systems architectures. As the major property of such organizations is information, protection of that information is essential. In a frequently changing operational infrastructure, security should be rapidly adapted into new organizational structures. Therefore, formal modeling leading into automation is required. As automation reduces the need for human, informal intervention, a model of protection of processing of requirements is required.

Second, the model provides evidence of the applicability of an underlying model for the management of information security. As one of the generic requirements for security models is provision of tools for analyzing the model itself [Bell, 1988], it is essential that scientific advances in the management of information security lead to models that can be formally analyzed to improve scientific and practical knowledge of the management process by identifying new areas of research and providing new types of analysis, such as formal approaches, of traditional research questions. One example of such traditional research questions not addressed before, is the provision of security for the management of information security. As has been shown in this paper, formal treatment of information security requirements enables exact specification of intuitive concepts, such as "management of information security" and "secure management of information security".

The fundamental question regarding applicability of the proposed model is the fundamental question of security models [Bell, 1988]: Formulation of those intuitive concepts always restricts their scope to meet requirements of formal reasoning. Therefore, it is essential that the proposed model does not unnecessarily restrict the scope of concepts under analysis leading into significantly reduced application scope of the model. As the model under analysis [Leiwo and Zheng, 1997a] is focusing on continuous processing of information security requirements, the approach taken herein does not restrict the applicability of the model, and hence is within the constraints specified by the original model. As has been argued before, treatment of the management of information security as a continuous process of requirement processing has several advantages, and hence the access control model does not unnecessarily restrict the scope of the management of information security.

An obvious question is whether access control is an adequate protection measure for security requirements. In distributed systems, a secure requirement exchange protocol is required to provide assurance of confidentiality and integrity of requirement during transformation, authenticity of units and non-repudiation of requirement exchange transactions. There is, anyhow, no significant difference between requirement exchange and any other communication between distributed units. Therefore, requirement exchange can be integrated into an existing communication framework and assumed to be secure. As formal proof of security is required only on systems with very high security requirements, an explicit assumption can be made that a secure communication infrastructure between units exists.

Acknowledgments Part of the work has been completed while Dr. Zheng was on sabbatical at the University of Tokyo. He would like to extend his thanks to Professor Hideki Imai for hospitality.

References

(1988). Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture. International Standard ISO 7498-2.

Bell, D.E. and La Padula, L.J. (1973). Secure computer systems: Mathematical foundations. Technical Report MTR-2547, vol. 1-2, Mitre Corporation, Bedford, MA, USA.

Bell, D. E. (1988). Concerning "modeling" of computer security. In *1988 IEEE Symposium on Security and Privacy*, pages 8–13.

Biba, K.J. (1977). Integrity considerations for secure computer systems. Technical Report MTR-3153, MITRE Corporation, Bedford, MA, USA.

Booysen, H.A.S. and Eloff, J.H.P (1995). A methodology for the development of secure application systems. In *IFIP TC11 11th International Conference on Information Security*.

Boswell, A. (1995). Specification and validation of a security policy model. *IEEE Transactions on Software Engineering*, 21(2):63–68.

Brinkley, D. L. and Schell, R. R. (1995). Concepts and terminology for computer security. In Abrams, Marshall D., Jajodia, Sushil, and Podell, Harold J., editors, *Information Security - An Integrated Collection of Essays*. IEEE Computer Society Press, Los Alamitos, CA, USA.

Denning, D.E. (1976). A lattice model of secure information flow. *Communications of the ACM*, 19(5).

Fisher, R. (1984). *Information Systems Security*. Prentice-Hall.

Harrison, M.A., Ruzzo, W.L., and Ullman, J.D. (1976). Protection in operating systems. *Communications of the ACM*, 19(8):461–471.

Jajodia, S., Samarati, P., and Subrahmanian, V. S. (1997). A logical language for expressing authorizations. In *IEEE Symposium on Security and Privacy*.

Leiwo, J. and Zheng, Y. (1997a). A formal model to aid in documenting and harmonizing information security requirements. In *IFIP TC11 13th International Conference on Information Systems Security*.

Leiwo, J. and Zheng, Y. (1997b). A framework for the management of information security. In *1997 Information Security Workshop*.

Parker, D. B. (1981). *Managers Guide to Computer Security*. Prentice-Hall, Inc, Reston, Virginia, USA.

Sandhu, R. S. (1993). Lattice-based access control models. *Computer*.

Sandhu, R. S., Coyne, E. J., Feinstein, H. J., and Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2):38–47.