A Lightweight Secure Monitoring Framework Based on Policies

WEN Hong-zi[1,2], QING Si-han[1,2], HE Jian-bo[1,2] ,ZHOU Yong-bin[1,3]

{1. Institute of Software;
2. Engineering Research Center for Information Security Technology;
3. State Key Laboratory of Information Security},
Chinese Academy of Sciences,
P.O. Box 8718 Beijing 100080, China
Email: wenhongzi@vip.sina.com; wenhongzi@msn.com

Abstract

The current secure monitoring facility has the problems of the redundancy of logging data and the delay of auditing. This paper proposes a formal Lightweight Secure Monitoring Framework Based on Policies (PB-LSMF), which not only can solve the previous problems but also adapts to strictly processing resources. Lastly, the application of Bell-LaPadula secure policies in PB-LSMF is discussed as an example.

Key words: secure monitoring, relation pattern, auditing, Bell-LaPadula secure policies,

1 Introduction

The secure monitoring, which is the most important one of the five audit targets proposed in [1] by American National Computer Security Center(NCSC), detects the system abnormities resulted from external penetrating and internal misusing by logging and analyzing secure events [2]. It provides confidence in an additional level for users that the system is in a secure state along with other auditing facilities [1]. However, the traditional secure monitoring share the common problems of the redundancy of logging data and the delay of detecting abnormities that Simone detailed in [3] with other auditing facilities. It will become more serious in WLAN/mobile environments, where the computing resources, such as processor performance and storage capacity, are very limited, than in LAN/host environments. Whereupon, it is vital to develop a lightweight secure monitoring mechanism to deal with the above difficulties.

On the other hand, a secure policy model considers all aspects of the system security, so it was usually regarded as a most abstract secure system [4][5]. Ideally, a system that is implemented according to its secure policy model can guarantee system security expected.

However, the system policies may not be implemented correctly and maintained appropriately because of the limitations of current techniques in developing and running a secure system, and thus it is necessary to additionally ensure that the policies correctly function by appropriate ways.

Based on the above facts, we propose a formal Lightweight Secure Monitoring Framework Based on Policies(PB-LSMF for short) [6]. It is not only has considerable security but also merely requires the minimal resources. Therefore, it can effectively address the problems of the traditional auditing activities.

The rest of this paper is organized as follows. The second section introduces the related work. The PB-LSMF is detailed in Section 3. The Section 4 discusses the application of Bell-LaPadula secure policy to PB-LSMF. The Section 5 analyzes the PB-LSMF, and the last section concludes our efforts.

2 Related works

The subject of this paper is mainly related to M. Bishop works [7,8,9]. In [7] he suggested to reduce logging data by the concept of the relevant part of the system state, and detect the errors by error-rejection mechanism. However, he didn't show how to associate the relevant part of the system state with the system components, and the content of the relevant part of the system state is decided by the experience of system administrators or developers. Furthermore, since the relevant part of the system state is only used to reduce logging data, it is no help to lower the redundancy of logging data. In his 1995 paper [8], although he recognized that the goal and content of secure monitoring are decided by system secure policies and its implementing mechanism, he didn't present the way how the system policies associate with the goal and content of secure monitoring yet. In [9], he formally discussed the relation among the system policies, the content of logging and the goal of secure monitoring, and proposed a Goal-Oriented Auditing and Logging Model. However, because a very abstract Turing machine was employed to model practice system, the mapping from secure policies to the goal of auditing is still very thorny.

In 1980, Anderson [10] first suggested to use audit trails to monitor the secure threats, and also proposed to develop simple auditing tools to check the existing logging data in order to find the invalid accesses to system and file. He didn't approve of altering the basic structure of logging system designs, so his proposals can neither release the system burden caused by logging nor significantly reduce the delay of auditing. Bonyun [11] regarded a single and well-formed logging procedure as a component of computer secure mechanism, and then discussed how auditors and secure officers decide logging content and extract the logging information from system. Picciotto [12] suggest the logging data should be reduced in order to find the abnormal trails in time. Mayfield [13] proposed to overcome the delay of auditing by monitoring and alarm.

Intrusion detection [14,15,16], which is a hot issue and closely related to secure auditing, shares the common ground on trying to find potential intrusion by analyzing the system logging data with the secure monitoring mechanism. However, the former mainly based on the general logging data, which means more logging data is desired, and tried to match all known intrusion pattern [14], or used the methods of probability and statistics to deduce the

potential intrusions from the logging data [15]; On the other hand, the secure monitoring framework decides the logging content according to certain monitoring targets and secure policies related to those targets firstly, and validates the part or whole system state. It is obvious that the latter can provide high performance, and accurate and timely results.

3 PB-LSMF

MP-SFM can be illustrated with Figure 1. It consists of four function components: deciding audit targets, deciding logging items, logging and monitoring. PolicyItem, AuditTarget, LogItem, SysData, LogData, SysSRP and Result are secure policy item, secure monitoring target, logging item, system data, logging data, set of SyaData dominated by a system policy (and then called as the relevant part of the system state) and the result of judgment, respectively. POLICYITEMs, AUDITTARGETs, LOGITEMs, SYSDATAs, SYSSRPs and RESULTs are severally set of PolicyItems, AuditTargets, LogItems, SysDatas, SysSRPs and Results. The functions and specifications of the PB-LSMF are detailed as following:

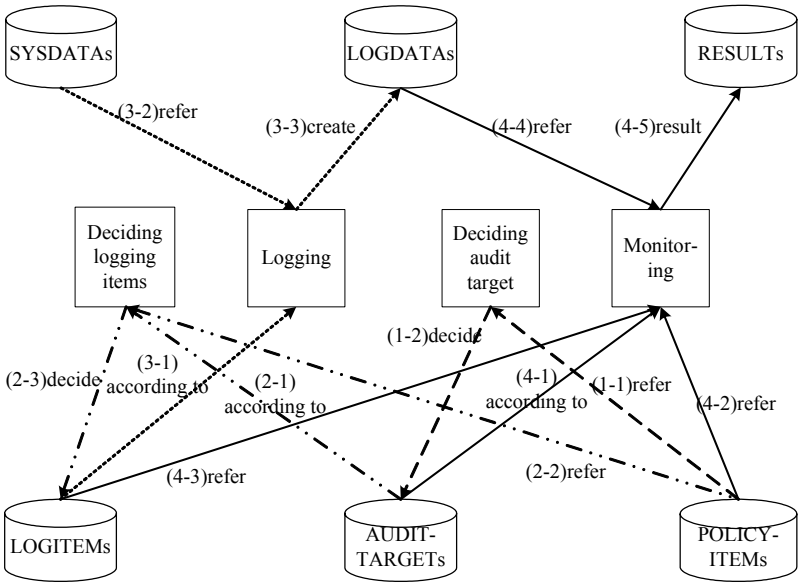

Figure 1: PB-LSMF diagram

3.1. Deciding audit targets

$$CreatAuditTargets : 2^{POLICYITEMs} \rightarrow AUDITTARGETs$$

Its function is to create AuditTarget(1-2) referring secure policies(1-1). As we know, the aim of secure monitoring is to provide a confidence in an additional level for users that the system secure policies are correctly implemented and effectively functioning, and ensures the system in a secure state. It is embodied by AuditTargets. The mapping from an AuditTarget to POLICYITEMs is defined as:

$$Map_{AuditTarget-POLICYITEMs} : AUDITTARGETs \rightarrow 2^{POLICYITEMs}$$

Here POLICYITEMs is the set of policies implemented in system, and the AUDITTARGETs are decided by referring to the elements in POLICYITEMs.

There is a SysSRP corresponding to each PolicyItem, and its mapping function is defined as:

$$Map_{PolicyItem-SysSRP} : POLICYITEMs \rightarrow SYSSRPs$$

Here if a PolicyItem∈POLICYITEMs, then its index in SYSSRPs is denoted as $\sigma_{PolicyItem}$.

3.2. Deciding logging items

$$CreatLogItems : AUDITTAGERTs \times 2^{POLICYITEMs} \rightarrow 2^{LOGITEMs}$$

This function is used to decide the content of logging. It determinates the LOGITEMs(2-3) based on the AUDITTARGETs(2-1) by referring to POLICYITEMs (2-2). The following is its specification:①

$$\begin{aligned} &CreatLogItems(lis : 2^{LOGITEMS}) \triangleleft \\ &\quad at : AuditTarget;\ p : PolicyItem \\ &\quad lis = \bigcup_{at \in AUDITTARGETs} \left(\bigcup_{p \in Map_{AuditTarget-POLICYITEMs}(at)} \{Map_{PolicyItem-SysSRP}(p)\} \right) \triangleright \end{aligned}$$

In the above specification, we compute the PolicyItems sets of each AuditTarget firstly, and then combine those sets denoted with LOGITEMs.

3.3. Logging

$$SysLog : 2^{SYSDATAs} \times LOGITEMs \rightarrow LOGDATAs$$

The function transforms SYSDATAs(3-2) to LOGDATAs(3-3), and the transformation content is decided by LOGITEMs (3-1).

In order to map system data to log data, the following function is defined:

$$Map_{LogItem-SysData} : LOGITEMs \rightarrow SYSDATAs$$

In multi-policy cases, different policy has different naming system and name space for the same system components, so a system component may have several names. This function mainly deals with the above naming problem and makes a system data constrained by several policies to be logged only once.

$$SingleLog : SYSDATAs \rightarrow LOGDATAs$$

This function transforms single SysData to single LogData. So the specification of system log function SysLog is:

$$\begin{aligned} &SysLog(lds : 2^{LOGDATAS}) \triangleleft \\ &\quad sd : SysData;\ li : LogItem \\ &\quad lds = \bigcup_{li \in LOGITEMs} \{SingLog(Map_{LogItem-SysData}(li))\} \triangleright \end{aligned}$$

3.4 Monitoring

$$AuditMonitor : AUDITTARGETs \times 2^{LOGDATAs} \times 2^{POLICYITEMs} \times 2^{SYSSRPs} \rightarrow RESULTs$$

Here RESULT={TRUE, FALSE}, TRUE denotes that a system is in a consistent state, and FALSE denotes that the system encounters an inconsistent state. TRUE and FALSE are Boolean value, and the operators employed on them obey Boolean algorithm. The function depends on a AuditTarget(4-1), refers PolicyItems related to this AuditTarget(4-2) and the LogItems corresponding SysSRPs of those policy(4-3), refers the SysDatas related to those SysSRPs(4-4) and then validates the system state(4-5).

To evaluate the system state of each policy, we introduce the following function:

$$Evalute : POLICYITEMs \times 2^{LOGDATAs} \rightarrow RESULTs$$

The following function models the mapping from LogItem to LogData:

$$Map_{LogItem-LogData} : LOGITEMs \rightarrow LOGDATAs$$

Based on the above auxiliary functions, the specification of monitoring is:

$$
\begin{aligned}
&AuditMonitor(\,decision : RESULT\,) \triangleleft \\
&\quad at : AuditTarget;\ p : PolicyItem;\ \ ss : SysSRP;\ lds : LOGDATAs \\
&\quad ss = Map_{PolicyItem-SysSRP}(p) \\
&\quad lds = \bigcup_{li \in ss} \{Map_{LogItem-LogData}(li)\} \\
&\quad decision = \bigwedge_{p \in Map_{AuditTarget-POLICYITEMs}(at)} Evalute(p,\ lds) \triangleright
\end{aligned}
$$

In the above specification, the part system state corresponding to each policy which relates to each AuditTarget is evaluated, and then the whole system state is validated. If the result is TRUE it means that the system state is valid, or else the system state is invalid and the result is false. Once the valid state is detected, the predefined system actions will be called to deal with those abnormalities.

4 The application of Bell-LaPadula secure policies to PB-LSMF

Since Bell-LaPadula secure policies(BLP for short) [6] is both the government secure policies and the military standard secure policies [18], we employ it as the example for detailing the application of secure policies to PB-LSMF.

In order to monitor the system state, the PB-LSMF will firstly apply the secure policies referenced by current AUDITTARGETs to the LOGDATAs, and then evaluate the system state. However, the current form of BLP policies is difficult to be straightway employed in the decision process of PB-LSMF which is mainly automated. Therefore, the first thing is to change polices form to a suitable one. In this paper, the formal relation pattern is chosen.

4.1 Relation and pattern

The set of system consistent statements defines the system consistent state, and then the secure state of system is defined by the set of system secure consistent statements. A statement is usually seen as a binary relation so that we can benefit from the solid mathematic foundation of the relation [19][20]. However, the secure relation and the secure policy are different thing, because the secure relation is related to state but polices are not. The policies cannot be used to validate the system state until they combine with the system state information related to those policies. In order to benefit from the above features of relations, the relation pattern, which has the same mathematical foundation as relations, is employed to rewrite the policy items. A secure policy can not form a corresponding secure relation until its pattern is combined with the system secure state information.

4.2 BLP relation patterns

In a system where BLP policies were implemented, the capability that a subject accesses an object is expressed by access capability (Capability for short). Capability is a triple of the form (Subject, Object, Op), in which Subject is subject, Object is object and Op is operation. It means that Subject has the permission to access Object with Op. SUBJECTs、OBJECTs、OPs、CAPABILITYs are the set of Subject, Object, Op and Capability, respectively. SecureLabel is a secure label with the form of 2-tuple (SecureLevel, SecureCategory), in which SecureLevel is secure level of SecureLabel and SecureCategory is secure category. The set of SecureLabels, SecureLevels and SecureCategorys are denoted as SECURELABLEs,

SECURELEVELs and SECURECATEGORYs, respectively. The SecureLabel related to subject is called as SubjectSL with the form of 2-tuple (Subject, SecureLabel), and ObjectSL is related to subject with the form of 2-tuple (Object, SecureLabel). SUBJECTSLs and OBJECTSLs are the set of SubjectSL and ObjectSL.

In order to select the value of specific item in a tuple, the **function for selecting general tuple item** is defined:

$$GetProperty : (Tuple, index) \rightarrow Tuple[index]$$

Here *Tuple* is the name of tuple, *index* the index value of tuple item whose value begins with 1, *Tuple[index]* the value of the *index*-th item of tuple. The function is used to select the value of specific item in tuple.

operation classification function：

$$OpClass : OPs \rightarrow OPCLASSs$$

which is used to discriminate the type of Op, where the set of Op types is OPCLASSs＝{Read, Write}. From the function, we know that an Op belongs to one and only one of two Op types, Read or Write.

The simple secure property of BLP (**BS** for short) can be rewritten as the following:

$$\begin{aligned}
&R_{BS}:\forall capability \in CAPABILITYs \\
&\quad \exists subject \in SUBJECTs, object \in OBJECTs, op \in OPs, subjectSC \in SUBJECTSCs, objectSC \in OBJECTSCs \\
&\quad [capability = (subject, object, op) \wedge \\
&\quad GetProperty(subjectSC, 1) = subject \wedge \\
&\quad GetProperty(objectSC, 1) = object \wedge \\
&\quad OpClass(op) = Read \wedge \\
&\quad GetProperty(GetProperty(subjectSC, 2), 1) \geq GetProperty(GetProperty(objectSC, 2), 1) \wedge \\
&\quad GetProperty(GetProperty(subjectSC, 2), 2) \supseteq GetProperty(GetProperty(objectSC, 2), 2)]
\end{aligned}$$

Here, if Op type is Read, the SecureLevel of SubjectSL dominates the one of ObjectSL, and the SecureCategory of SubjectSL contains the one of ObjectSL.

The relevant part of the system state of BS is:

$$\sigma_{BS} = \{SUBJECTs, OBJECTs, OPs, SECURELEVELs, SECURECATGORYs\}$$

The *-property of BLP (**B*** for short) can be rewritten as the following:

$$\begin{aligned}
&R_{B*}:\forall capability \in CAPABILITYs \\
&\quad \exists subject \in SUBJECTs, object \in OBJECTs, op \in OPs, subjectSC \in SUBJECTSCs, objectSC \in OBJECTSCs \\
&\quad [capability = (subject, object, op) \wedge \\
&\quad GetProperty(subjectSC, 1) = subject \wedge \\
&\quad GetProperty(objectSC, 1) = object \wedge \\
&\quad OpClass(op) = Write \wedge \\
&\quad GetProperty(GetProperty(subjectSC, 2), 1) \leq GetProperty(GetProperty(objectSC, 2), 1) \wedge \\
&\quad GetProperty(GetProperty(subjectSC, 2), 2) \subseteq GetProperty(GetProperty(objectSC, 2), 2)]
\end{aligned}$$

Here, if Op type is Write, the SecureLevel of SubjectSL is dominated by the one of ObjectSL, and the SecureCategory of ObjectSL containes one item of SubjectSL.

The relevant part of the system state of B* is:

$$\sigma_{B^*} = \{SUBJECTs, OBJECTs, OPs, SECURELEVELs, SECURECATGORYs\}$$

4.3 The application of BLP relation patterns to PB-LSMF

When system only enforced the BLP, the POLICYITEMs of PB-LSMF that consists of the items of BLP relation patterns, and the secure monitoring targets will be resulted from the elements of {BS, B*}. SYSSRPs , SYSSRPs＝$\{\sigma_{BS}, \sigma_{B^*}\}$, consists of the relevant part of the system state of BLP relation patterns. The relevant part of the system state of PolicyItem, PolicyItem∈POLICYITEMs, in SYSSRPs can be written as $\sigma_{PolicyItem}$, for example, the relevant part of the system state of BS is σ_{BS}.

5 Analyses

5.1 PB-LSMF integrity analysis

The integrity of PB-LSMF means that the current monitoring targets set can decide the state of whole system. It is embodied as proposition 1:

Proposition 1: The system can be in a consistent state *if and only if*

(1) $\bigcup_{at \in AUDITTARGETs} Map_{AuditTarget-POLICYITEMs(at)} = POLICYITEMs$

(2) $[\forall at \in AUDITTARGETs$

$$[\bigwedge_{p \in Map_{AuditTarget-POLICYITEMs}(at)} Evalute(p, (\bigcup_{li \in Map_{PolicyItem-SysSRP}(p)} \{Map_{LogItem-LogData}(li)\}) = TRUE]]$$

Proof: (if) only (1) when the union of the secure policies sets referred by each system monitoring target equals POLICYITEMs, it is certain that the current monitoring targets can reflect the whole system state; (2) when all of policies in each target can be satisfied, the relevant part of the system state related to each monitoring target is consistent. Together with (1) and (2), the whole system is consistent.

(only if) According to the definition of secure system state, it is necessary that the current state information satisfies all policy relation patterns when the system state is consistent. It means that: (1) every policy must be referenced by a monitoring target at least once. In other words, the set of system secure policies mapped from monitoring targets set is fulfilled;(2) all of the policies included in each monitoring target is satisfied. □

Corollary 1: In PB-LSMF, if there is one of monitoring targets that cannot be satisfied, the system must be inconsistent; Although all of monitoring targets are satisfied, the system might not be consistent unless the (1) condition of proposition 1 holds.

5.2 Performance

In traditional secure systems, although logging is automated, auditing used to analyze logging data is by hand. In order to reach considerably secure assurance with auditing, the data needed to be logged as far as possible. As a result, the system has to bear heavy burden of performance and store space. Furthermore, along with the logging data increasing, the system abnormities might not be detected or be detected too later.

Through improving the efficiency of logging data usage and adapting automate way to auditing the logging data, PB-LSMF successfully solve the above problems. Firstly, by selecting minimal logging data set, which can satisfy the certain monitoring targets and in

which there is no redundant data, the efficiency of logging data usage is ultimately improved. Secondly, after the system policies are rewritten as corresponding relation patterns, the policies can be directly adapted to the automated auditing process, and thus the delay of auditing can be reduced.

Configurable feature is another advantage of PB-LSMF. If we don't care about the whole system consistency, we can decide a special part system state according to certain monitoring targets.

From the above analysis we can safely draw the conclusion that the PB-LSMF has the same characteristics such as relatively few processing resources and low delay. Especially, the configurable feature can further optimize its performance. Those features can make PB-LSMF adapt to stricter computing resources.

5.3 Experiment and Result Analysis

We have implemented the secure monitor framework on the ERCIST B2 secure operating system, which is developed by Engineering Research Center for Information Security Technology, Chinese Academy of Science. The ERCIST secure operating system is a multi-policy operating system, which implements BLP, RBAC and DTE security policy, and is a kind of security enhanced Linux. The secure monitoring is used to monitor the effectiveness of BLP MLS policy. To validate the target, the least set of log items is adopted. To be explicit, the logs resulting form no secure policy being enforced, and the logs from all secure policy being enforced are contrasted and analyzed. We denote 1000 systems call as a metric unit. The effects of audit operations on the response to system calls and on the memory consume are evaluated respectively. The results are shown in table 1-table 3. The table 1 shows some performance parameters of the secure OS when the log audit system is disabled. Table 2 is the performance results when the least set of log items of the system is audited. The performance parameters when all of log items are audited are showed in table 3.

Number	Run time(μs)	Used disk space (k)
1	24341	0
2	33563	0
3	24781	0
4	24745	0
5	24718	0

Table 1: the experimental results when the Log Audit system is disabled.

Number	Run time(μs)	Used disk space (k)
1	41810	99.4
2	61204	120.4
3	42210	99.8
4	50353	103.7
5	42150	87.1

Table 2: the experimental results when the Least set of is audited.

Number	Run time (μs)	Used disk space (k)
1	219494	201252
2	236375	216332
3	291772	253235
4	224468	202154
5	273609	257192

Table 3 the performance parameters when all log items are audited

Form the table 1, 2 and 3, we can see that the MP-SMF improves the utility of log data, reduces the effect that audit subsystem makes on the whole system, and enhances the performance of the whole secure operating system.

6 Conclusions

In order to detect the system state, secure monitoring, as a main goal of system auditing, detects various outside penetrates and inside misuses by recording and checking system secure events. The goal of the secure monitoring is represented as the secure monitoring targets mainly decided by the system secure policies. This paper proposes a formal secure monitor framework based on policies (PB-LSMF). It not only can solve the problem that the redundancy of logging data and the delay of auditing but also has configurable features. Those features can make PB-LSMF adapt to stricter computing resources. Finally, the application of Bell-LaPadula secure policies in PB-LSMF is discussed as an example.

References

[1] National Computer Security Center, *A Guide to Understanding Audit in Trusted Systems*, Version 2, NCSC-TG-001, Fort George G. Meade, MD, 1988.

[2] K. Seiden, J. Melanson, *The Auditing Facility for a VMM Security Kernel*, IEEE Symposium on Security and Privacy, Oakland CA, May 1990, pp.2-19.

[3] F. H. Simone, *IT-Security and Privacy*, Springer, 2001, pp.35-104.

[4] U.S. Department of Defense, *A Guide to Understanding Security Modeling in Trusted Systems,* Tech. Rep. NCSC-TG-010, U. S. National Computer Security Center, Oct. 1992.

[5] M. Gasser, *Building A Security Computer System,* Van Nostrand Reinhold, 1988.

[6] D. E. Bell and L. J. LaPadula, *Secure Computer Systems: Mathematical Foundations and Model*, Technical Report M74-244, The MITRE Corporation, Bedford, MA 01730, 1974.

[7] M. Bishop, *A Model of Security Monitoring*, IEEE 5th Annual Computer Security Applications Conference, December 1989, Tucson Arizona, pp.46-52.

[8] M. Bishop, *A Standard Audit Trail Format*, Proceedings of the Eighteenth National Information Systems Security Conference, Oct. 1995, pp. 136-145.

[9] M. Bishop, C. Wee, J. Frank, *Goal-Oriented Auditing and Logging*, http://seclab.cs.ucdavis.edu/papers/tocs-96.pdf, 1996.

[10] J. P. Anderson, *Computer Security Threat Monitoring and Surveillance*, Tech. Rep., James P Anderson Co., Fort Washington, PA, Apr. 1980.

[11] David Bonyun, *The Role of a Well-Defined Auditing Process in the Enforcement of Privacy Policy and Data Security*, IEEE Symposium on Security and Privacy, Oakland CA., 1981, pp.19-26.

[12] J. Picciotto, *The Design of An Effection Auditing Subsystem*, IEEE Symposium on Security and Privacy, Oakland CA., 1987, pp.13-22.

[13] T. Mayfield, J. E. Roskos, S. R. Welke, J. M. Boone, *Integrity in Automated Information Systems,* Tech. Rep, 79-91, NCSC, 1991.

[14] W. Lee and S. J. Stolfo, *Data Mining Approaches for Intrusion Detection*，In Proceedings of the 7th USENIX Security Symposium, San Antonio, Texas, U.S.A., 1998, pp.79-94.

[15] T. Carla, E. Brodley, *Temporal Sequence Learning and Data Reduction for Anomaly*

Detection, In Proceedings of the 5th Conference on Computer &Communications Security, New York, ACM Press, 1998, pp.150-158.
[16] S. Kumar, *Classification and Detection of Computer Intrusions*, PhD Dissertation, Purdue University, August 1995.
[17] J. Woodcock, J. Davies. *Using Z.* Prentice Hall, 1996.
[18] U.S. Department of Defense, *Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, National Computer Security Center, Fort Meade, MD, 1985.
[19] E. F. Codd, *Relational Databases: A Practical Foundation for Productivity*, Commun. ACM, Vol. 25, No. 2, 1982, pp.109-117.
[20] M. Tamer özsu, P. Valduriez, *Principle of Distributed Database Systems* (2th Ed.), Prentice Hall, 1989, pp.25-51.

① The notation used in the formal specification of requirements is basically a subset of the Z notation [17]. The only major change is the representation of a schema:

Schema - *Name* (*Declaration*) ◁ *Predicate*; ⋯ ; *Predicate* ▷

Top