An Analysis of the Chinese Wall Pattern for Guaranteeing Confidentiality in Grid-based Virtual Organisations

G. Dallons P. Massonet J.-F. Molderez C. Ponsard A. Arenas

Abstract—Virtual organisations (VO) allow independent organisations to share resources and collaborate to achieve common goals. When a VO is defined in a business context and confidential information is shared, security becomes a main concern. Furthermore, business contexts the VO need to adapt quickly to changes in the environment. Maintaining security in a dynamic environment is particularly challenging open issue in current Grids. The security issues are not only related to the protection from the outside world but also to the protection of the integrity and confidentiality of each organisation against potentially conflicting goals within the VO.

To tackle this problem, this paper shows how the general Chinese wall security model can be adapted for use in Grid-based VO. The result is a reusable Chinese wall pattern that is expressed in terms of a VO ontology. The pattern formalization is also proved using the Alloy SAT-based technology. This work is a first step towards deploying Chinese walls in operational Grid-based VO.

Index Terms—Virtual organisation, Security, Trust, Chinese Wall, Verification, Alloy, KAOS

1 Introduction

Virtual organisations (VO) allow resources to be shared across administrative domains, and thus enable collaborations between organisations. When VO operate in a business context the collaborations between organisations involve the exchange of confidential information, and thus have strong security requirements. Furthermore most business collaborations are very dynamic and need to adapt quickly to changes in the environment or in the goals of the collaboration. The confidentiality requirements thus need to be guaranteed in a dynamic environment. Guaranteeing these kind of security requirements in Grids is a very challenging issue with the current Grid security mechanisms. Currently the security mechanisms in the Grids that underlie VO are considered too static to support the dynamic security requirements that business collaborations have.

In this work, we investigate one security policy for enforcing confidentiality dynamically called the Chinese wall model. It is based on the prevention of conflicts of interest that arise during the lifetime of a system and are do not exist in the initial state of the system. The Chinese wall model deals with this problem by dynamically defining the access rights: the access decision is not only based on the rights of subjects, but is based on the history of past accesses. The Chinese wall security problem is well-known in the financial world. Some papers have addressed this topic and have given formal models to ensure that a Chinese wall can be maintained in a financial company [1], [5].

In this paper, we consider how to model the Chinese wall security policy in terms of VO, and define a pattern that can be used for defining confidentiality in VO. The definition of a Chinese wall pattern for VO, is a first step towards deploying Chinese walls in operating VO.

The paper is organised as follows: to handle this problem in the particular context of VOs, we introduce virtual organisation concepts in section 2. Then, we give a definition of the Chinese wall policy (section 3), and explain how a pattern can be modelled using the KAOS goal oriented approach (section 4). We then present a way to verify the correctness of this pattern (section 5). Then, we conclude with a discussion on this approach and future work (section 6).

2 VIRTUAL ORGANISATION CONCEPTS

To consider security in VO, it is important to reason on a common conceptual framework. In this section, we introduce the main concepts of our VO metamodel in order to share a common vocabulary [10].

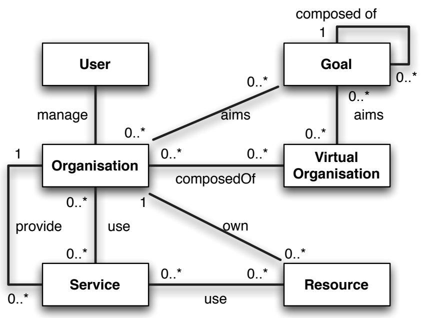

Figure 1. Virtual Organisation metamodel

The VO metamodel is presented in figure 1. We consider that a virtual organisation is a composition of organisations. An organisation is a company or an administration that plays a role in the VO. For example, an organisation owns a resource used by a service of an organisation participating to the VO. An organisation owning a resource has the right to add or remove the resource. A resource is an element used by a service to meet its specifications. A resource is owned by an organisation and is necessary to the execution of the service. Resources can be further refined into a taxonomy, such as distinguishing hardware and software resources. In the Chinese wall context, resources of interest are data.

A VO is formed for a purpose. The purpose of the VO can be captured by one or several high-level goals. The high-level goals can progressively be refined into more operational goals, resulting in a goal refinement structure. The VO goals can be met by the composition of finer goals that are assigned to individual organisations. Organisations manage their users which have access to the services used by their organisation.

3 THE CHINESE WALL

The Chinese wall addresses problems of confidentiality and access control. This problem is well-known and has been presented in several papers [1], [5]. In this article, we use the Chinese wall model from [1]. To promote reuse within the design of Grid systems, we have adapted the terminology and to fit the discussion within the space allowed for this paper, we will only focus on read access. There are several concepts appearing in the Chinese wall model (figure 2):

- A **resource** is an individual item of information belonging to an organisation.
- An **organisation** has several resources used in the context of her services. This set of resources is represented by the ″owned″ relationship between organisation and resource.
- Different **organisations** can be in competition. In this case, each of them belongs to the same **conflict of interest class**.
- A **user** is someone who makes an **access** to a resource through a **service** provided by an organisation.

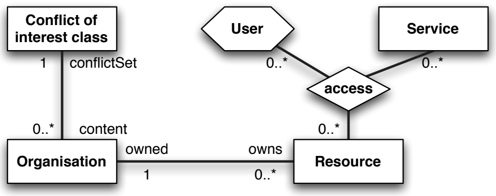

Figure 2. Object model

The Chinese wall is based on the idea that you can't access a conflicting resource. A conflicting resource is a resource accessed by a user that belongs to an organisation that is in the same conflict of interest class than other organisation resources accessed by this user.

Brewer and Nash have defined two security rules ensuring that the Chinese wall is not violated by wrong access [1]. They said that access by a user is only granted if the resource requested:

- is either owned by the same organisation than a resource already accessed by that user

- or belongs to an entirely different conflict of interest class.

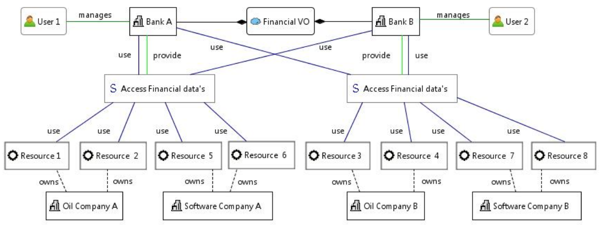

Figure 3. Example of virtual organisation

We will consider the following example of VO (figure 3). This example presents a Financial VO composed by two banks: Bank A and Bank B. Each of these Banks provides a service to access financial data and has access to the service of the other bank. The financial data is managed by resources 1 to 8. Each of these resources is owned by an organisation which is a customer of the bank.

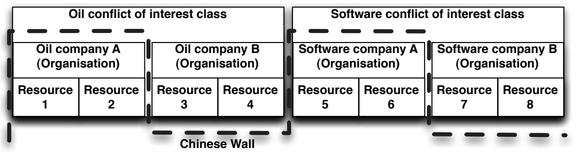
Oil conflict of interest class
Software conflict of interest class
Oil company A (Organisation)
Oil company B (Organisation)
Software company A (Organisation)
Software company B (Organisation)
Resource 1
Resource 2
Resource 3
Resource 4
Resource 5
Resource 6
Resource 7
Resource 8
Chinese Wall

Figure 4. Example of a chinese wall

To explain the Chinese wall in the context of this example, we add the figure 4 representing the conflict of interest classes. If "user 1" has accessed resource 1, this user can only access the other resources of Oil Company A or resources form Software Companies since they belong to another conflicting class than Oil Company A. If the user later accesses resource 5 of Software Company A then its possible accesses will be restricted to Oil Company A and Software Company A since Software Company B is in a conflicting set with Software Company A. The Chinese wall is represented by the dashed line. An important remark is the dynamic aspect of the wall. Indeed, the wall changes according the history of accesses.

This example shows a situation where the Chinese wall problem exists in the VO but not for each organisation. The two conflict of interest classes exist for the financial VO because Oil Company A is a competitor of Oil Company B and Software company A is a competitor of Software Company B. If we take singly Bank A and Bank B, they don't need a Chinese wall. But if we take the VO, the Chinese wall is required to ensure that there won't be conflicting access.

4 Chinese Wall Pattern

The Chinese wall pattern is presented using KAOS [6], [11], [12]. KAOS is a goal-oriented method for requirements engineering. The method is based on the identification of goals which have to be met by the designed system. Goals are refined in subgoals. These subgoals imply the parent goal and are more detailed. Goals are refined until they can be operationalized -here through a policy- and assigned to agents. Goals can be formalized using linear temporal logic (LTL) [7]. Verifications can then be made on goal refinements to ensure that the system meets the goals and that the goal model is well-formed.

The chinese wall goal model is expressed in figure 5. It is formalized in a generic way using the previously introduced VO concepts and the notion of conflict of interest classes. This ensures that this description is a reusable pattern that can be instantiated to all VO.

The main goal is to ensure that there won't be a conflict during the life of the system. A conflict will occur if a user accesses a resource that is in the same conflicting class and owned by a different organisation than a resource previously accessed. So, the following goal must hold if we want to maintain the Chinese wall:

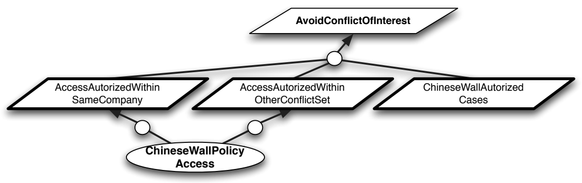
AvoidConflictOfInterest
AccessAutorizedWithin SameCompany
AccessAutorizedWithin OtherConflictSet
ChineseWallAutorized Cases
ChineseWallPolicy Access

Figure 5. Goal pattern for the Chinese wall

Goal AvoidConflictOfInterest
FormalDef $(\forall u{:}User; r, r'{:}Resource)\ noConflict(u, r, r')$

We will consider a writing shortcut $\exists_1^0\ o : Object$ which means that there exists zero or one o of type $Object$. This allows us to simplify the formula by avoiding huge disjunctions. The $\blacklozenge A$ formula means that A has been true sometimes in the past and the $\circ A$ formula tells us that A is true in the following state of the system. Let us also define some constrained relationships:

- hasAccessed: $(\forall u{:}User; r{:}Resource)\ hasAccessed(u, r) \Leftrightarrow \blacklozenge access(u, r)$
- sameOrganisation: $(\forall r, r'{:}Resource)\ sameOrganisation(r, r') \Leftrightarrow owned(r) = owned(r')$
- differentConflictSet: $(\forall r, r'{:}Resource)\ differentConflictSet(r, r') \Leftrightarrow conflictSet(owned(r)) \neq conflictSet(owned(r'))$
- noConflict: $(\forall u{:}User; r, r'{:}Resource)\ noConflict(u, r, r') \Leftrightarrow hasAccessed(u, r) \wedge hasAccessed(u, r') \wedge (sameOrganisation(r, r') \vee differentConflictSet(r, r'))$

To refine this goal, we will use the case-driven pattern refinement tactic in [2]. This tactic consists to split a goal into cases. So, if we have a goal $P \Rightarrow Q$, we can split it into three subgoals: $P \wedge C_1 \Rightarrow Q$, $P \wedge C_2 \Rightarrow Q$ and $P \Rightarrow C_1 \vee C_2$. In our case the parent goal has not the form prescribed by the pattern, more over we introduced several quantifiers: so we cannot take the refinement as proven and have to check for its correctness.

The first subgoal of the goal *AvoidConflictOfInterest* can be formulated as

Goal AccessAutorizedWithinSameCompany
FormalDef $(\forall r{:}Resource; \exists_1^0\ u{:}User; \exists_1^0\ r'{:}Resource)$ $hasAccessed(u, r) \wedge sameOrganisation(r, r') \Rightarrow \circ(\ hasAccessed(u, r'))$

If we access a resource in the same organisation than a resource already accessed, we are allowed to access. The second case is quite similar. We can access a resource that is in a different conflict of interest class than other resources accessed:

Goal AccessAutorizedWithinOtherConflictSet
FormalDef $(\forall r{:}Resource; \exists_1^0\ u{:}User; \exists_1^0\ r'{:}Resource)$ $hasAccessed(u, r) \wedge differentConflictSet(r, r') \Rightarrow \circ(hasAccessed(u, r'))$

The third element of the refinement expresses the case composition. It is defined as:

Goal ChineseWallAutorizedCases
FormalDef $(\forall u{:}User; r, r'{:}Resource)$ $hasAccessed(u, r) \Rightarrow (sameOrganisation(r, r') \vee differentConflictSet(r, r'))$

The access policy that operationalizes the Chinese wall can be specified by the following formula:

Policy ChineseWallPolicyAccess(u:User,r:Resource)
Rule 1 $conflictingResource(u, r) \Rightarrow skip$
Rule 2 $\neg conflictingResource(u, r) \Rightarrow hasAccessed(s, o)$

Where $conflictingResource(u, r)$:

$$(\forall u{:}User; r{:}Resource)\ conflictingResource(u, r) \Leftrightarrow (\exists r'{:}Resource) hasAccessed(u, r') \wedge (\neg(sameOrganisation(r, r')) \wedge \neg(differentConflictSet(r, r')))$$

Skip means that the relation $hasAccessed$ stays unchanged.

5 Verification with Alloy

In this section, we will verify that the pattern presented in the previous section is correct. The verification will be done using the Alloy analyser [4]. Once the pattern refinement has been checked it can be reused without having to perform the verifications again. However if the pattern cannot be reused as such, and needs to be adapted, then the adapted pattern will need to be checked again.

An important remark is the translation of temporal logic. In Alloy, there aren't temporal operations. We have introduced a state signature to model the time (listing 1). An always operator becomes an "all state" quantification and the next operator is translated by a "next[state]". The conceptual model presented in figure 2 is translated in Alloy (see listing 1). Each entity is translated as a "sig" signature in Alloy and the relations become attributes of these signatures.

Goals are refined into subgoals. The refinement is correct if it's complete and consistent. A desired refinement property is also the minimality but it is not required. In [3], a formal definition is given. A set of goals $\{G_1, G_2, \ldots, G_n\}$ refines a goal G in the domain D if the following conditions hold :

$$G_1, \ldots, G_n, D \models G \qquad \text{(completeness)} \quad (1)$$

$$\textstyle\bigwedge_{j \neq i} G_j, D \not\models G \text{ for each } i \in [1..n] \qquad \text{(minimality)} \quad (2)$$

$$G_1, \ldots, G_n, D \not\models false \qquad \text{(consistency)} \quad (3)$$

Listing 1. Alloy state model and conceptual model

```
module models/chinesewall
open util/ordering[State] as ord

//————————— State signature —————————
sig State {accessed : User—>Resource}

// No access at the initial state
fact initialState {let s0 = first[] | no s0.accessed}

// no retrictive access
pred access(state : State, next : state, u : User, r : Resource)
{next.accessed=state.accessed+u—>r}

// state transition
fact stateTransition {all state: State, next : next[state]|
one u : User, r : Resource{
next.accessed = state.accessed or access[state,next, u, r]}}

//——————————————— Model ———————————————
sig User {}
sig Resource {owned : one Organisation}
sig Organisation {conflictSet : one ConflictSet}
sig ConflictSet {}
```

As previously mentioned, the refinement must be complete and consistent. If it is possible, it can reach the minimal form. We have to check propositions 1, 2 and 3. To perform these checks, we have to specify an access operation that allows all accesses. The aim is to show that if the subgoals are met, then the main goal must hold (listing 2). So the accesses don't have to constrain the check.

The check of consistency is performed using a predicate and asking Alloy to give us an instance. If an instance can be found, then the predicate is consistent. The predicate is thus

$$AccessAutorizedWithinSameCompany \wedge AccessAutorizedWithinOtherConflictSet \wedge ChineseWallAutorizedCases$$

In our situation, Alloy found an example. This confirms that our refinement is consistent.

The completeness claim is checked differently using an assert. If the assert holds, no counterexample will be found. The assert in Alloy for the completeness is:

$$AccessAutorizedWithinSameCompany \wedge AccessAutorizedWithinOtherConflictSet \wedge ChineseWallAutorizedCases \Rightarrow AvoidConflictOfInterest$$

The analyser found no counterexample. So, the refinement is complete.

The minimality is checked by three asserts:

Listing 2. Main goal and refinement

```
//———————— Main Goal : AvoidConflictOfInterest ————————
pred conflict(state : State, r : Resource, r' : Resource,
u : User, u' : User){u—>r in state.accessed and
u'—>r' in state.accessed and u=u' and not(r=r') and
not(r.owned = r'.owned) and
(r.owned.conflictSet = r'.owned.conflictSet)}

pred AvoidConflictOfInterest(){all state : State|
all u : User | all r : Resource | all r' : Resource {
not(conflict[state,r,r',u,u])}}

//——————————— Refinement ———————————
// Predicate differentConflictSet
pred differentConflictSet(r : Resource, r':Resource)
{not(r.owned.conflictSet = r'.owned.conflictSet)}

// Predicate sameOrganisation
pred sameOrganisation(r : Resource, r':Resource)
{r.owned = r'.owned}

// Refinement
pred AccessAutorizedWithinSameCompany(){
all state : State| lone u : User | all r : Resource |
lone r': Resource {state != last[] and
u—>r in state.accessed and sameOrganisation[r,r']
implies u—>r' in next[state].accessed }}

pred AccessAutorizedWithinOtherConflictSet(){all state : State|
lone u : User | all r : Resource |
lone r' : Resource {state != last[] and u—>r in state.accessed
and differentConflictSet[r,r'] implies u—>r' in next[state].accessed }}

pred ChineseWallAutorizedCases(){all state : State|
all u : User | all r : Resource | all r' : Resource {
(state != last[] and u—>r in state.accessed )
implies sameOrganisation[r,r'] or differentConflictSet[r,r']}}
```

- $AccessAutorizedWithinSameCompany \wedge AccessAutorizedWithinOtherConflictSet \Rightarrow AvoidConflictOfInterest$
- $AccessAutorizedWithinSameCompany \wedge ChineseWallAutorizedCases \Rightarrow AvoidConflictOfInterest$
- $AccessAutorizedWithinOtherConflictSet \wedge ChineseWallAutorizedCases \Rightarrow AvoidConflictOfInterest$

These asserts must be invalid. So each of them must give a counterexample. Our refinement is not minimal because only the first assert gives a counterexample. This is not important because the minimality is desired but not required.

The refinement is valid. The policy enforcement, presented at the end of the previous section, can be refined in a similar way and will not be detailed here. This policy is complete and consistent.

In this section, we have verified the refinement and give the result of the checks for the Chinese wall policy. Our model is correct and consistent. These verifications give us confidence in the Goal pattern corresponding to the Chinese wall problem.

6 Conclusion and future work

This study shows us that in the context of VO, a Chinese wall can be maintained if the VO is aware of the resources used through the services. Thus, it is important to model these elements when we consider security in VO. This work has an indirect impact on the thinking done in the context of the VO metamodel [10].

This paper has showed that security policies in the context of VOs can be checked with state of the art analysis tools like Alloy. These checks result in higher quality requirements that are consistent and complete. The examples and counterexamples found by the analyser can be used in later phases of the development (for example to derive test directives).

The approach adopted in this paper has limitations. Indeed, all verifications are done in a finite scope of state. The Alloy method uses the hypothesis that most of the problems can be found in a small scope. In practice, this hypothesis seems to be verified. The aim of this approach is to increase the confidence in the correctness of the pattern. As of result of the analysis presented in this paper we have high confidence in the correctness of the pattern, but do not claim it is correct due to the nature of the Alloy analysis technique used.

This work is realized in the context of two European projects on grids: GridTrust [8] and BeInGrid [9]. The two projects have to identify trust and security patterns and create a repository of these patterns. Our work is strongly link with Gridtrust and has an impact for BeInGrid. Indeed, the approach used can be reusable for other patterns. Further work will be to check other security patterns in the context of GridTrust. This work is also relevant for the SOA community which has considered the Chinese wall policy [13], [14].

The next step will be to consider the differences between read accesses and write accesses. Indeed, in some situations [1], the Chinese wall can be broken. In order to maintain the wall in the case of write access, we need information about the source of data written. Another extension is to consider the aggressive Chinese wall model [5].

Cases study will be analyzed in the GridTrust project. Patterns will be used in concrete situation and will be empirically validated. Probably, some patterns combinations will occur. In this case, verifications can be done to check if the properties of the patterns hold and if there aren't side effects.

Acknowledgement

This work is financially supported by the European FP6-2005-IST-5 GridTrust European project (project reference number 033817) and the Walloon Region (DGTRE).

References

[1] D. F. C. Brewer and M. J. Nash. The Chinese Wall Security Policy. In *IEEE Symposium on Security and Privacy*, pages 206-214, 1989.
[2] R. Darimont and A. van Lamsweerde, Formal Refinement Patterns for Goal-Driven Requirements Elaboration, *4th FSE ACM Symposium, San Francisco*, 1996.
[3] R. Darimont, Process Support for Requirements Elaboration, PhD Thesis, Université catholique de Louvain, Dépt. Ingénierie Informatique, Louvain- la-Neuve, Belgium, 1995.
[4] Daniel Jackson, Software Abstractions, *ISBN 0-262-10114-9, MIT Press*, 2006.
[5] T.Y. Lin, Chinese wall security policy-an aggressive model. In *Computer Security Applications Conference*, pages 282-289, 1989.
[6] Emmanuel Letier, Goal Oriented requirements Engineering with KAOS (Thesis), *Université catholique de Louvain*, 2001.
[7] Z. Manna and A. Pnueli, The Reactive Behavior of Reactive and Concurrent System, Springer-Verlag, 1992.
[8] GridTrust European research project, http://www.gridtrust.eu
[9] BEinGRID European research project, http://www.beingrid.eu
[10] P. Massonet, A. Arenas, Deliverable D1.1 : Survey of Grid Application Classes and VO Topologies, GridTrust Project
[11] The Objectiver Tool, http://www.objectiver.com
[12] The FAUST toolbox, http://faust.cetic.be, 2004
[13] Edgar Weippl and Alexander Schatten and Shuaib Karim and A. Min Tjoa, SemanticLIFE Collaboration: Security Requirements and Solutions - Security Aspects of Semantic Knowledge Management,PAKM, 2004, 365-377.
[14] Simon Foley, Stefano Bistarelli, Barry OSullivan, John Herbert, Garret Swart, Multilevel Security and Quality of Protection,
Proceedings of Quality of Protection, 2005