Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 197 (2008) 45–56
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Managing Conflicts of Interest in Virtual Organisations

Alvaro Arenas Benjamin Aziz Juan Bicarregui
Brian Matthews

e-Science Centre
STFC Rutherford Appleton Laboratory
Chilton, Didcot OX11 0QX, United Kingdom

Abstract

In this paper, we present a formal model of virtual organisations that incorporates the concept of conflicts of interest. The model, which follows an incremental development approach using Event-B, focuses on goals and organisations at the abstract level and introduces resources at the concrete level. The model is motivated by the type of virtual organisations used in the domain of scientific experiments. Individual organisations, at the abstract level, are allowed to pursue conflicting goals within a virtual organisation. However, at the concrete level, these conflicts are isolated by applying a separation of resources mechanism. This ensures that no resource is allocated to any two conflicting goals.

Keywords: Security, Virtual Organisations, Chinese Walls, Refinement

1 Introduction

Virtual Organisations (VOs) provide an abstraction to represent inter-organisational collaborations, a topic of fresh interest given the current exploitation of Internet technology to create virtual enterprises [9], or the sharing of resources across different organisations as envisaged by Grid computing [11]. A VO can be seen as a temporary or permanent coalition of geographically dispersed organisations that pool resources in order to achieve common goals.

In order to support rapid formation of VOs, it is necessary that the potential partners are ready and prepared to participate in such collaboration. This readiness includes common interoperable infrastructure, common operating rules, and common representation of capabilities, among others. The concept of Virtual Breeding Environment (VBE) has emerged as the necessary context for the creation of VOs [8]. A VBE can be defined as an association of organisations adhering to a base long term cooperation agreement and adoption of common operating principles and infrastructure with the main objective of participating in potential VOs.

doi:10.1016/j.entcs.2007.12.016

In this paper, we have adopted the view that potential partners in a VO are selected from a VBE. We are interested in goal-oriented VOs, so organisations willing to participate in a VO will join the VBE, advertising the goals they can achieve and their resources. We are assuming that this step has been performed previously and concentrate on the management of VOs.

VO architects should be able to evaluate at design time the likely consequence of the decisions that they make regarding VO architecture and policies. This paper contributes to this end by providing a method to analyse conflicts of interest in VOs. In our model, a VO could include conflicting goals. Organisations are allowed to work on conflicting goals as long as they do not allocate the same resource to two conflicting goals.

The rest of the paper is structured as follows. In Section 2 we discuss the problem of conflict of interest in VOs, and in particular within the domain of scientific experiments. In Section 3, we review the Event-B refinement methodology and give a quick guide on the language. In Section 4, we present a model of VOs based on goals and organisations such that it registers the requirement that conflicting goals within an organisation must be isolated. In Section 5, we present the concrete model, which contains also resources and ensures that the requirement in the abstract model is realised through the mechanism of the separation of resources. In Section 6, we discuss related work and finally, in Section 7, we conclude the paper and give directions for future work.

2 Conflicts of Interest in Virtual Organisations

A VO is created when an organisation wants to achieve a set of goals but it does not possess all the resources/capabilities needed to do so. The VO management infrastructure would look for potential partners capable of achieving the VO goals. In our model, a VO could include conflicting goals; furthermore, an organisation can participate in conflicting goals. However, the VO management infrastructure restricts such organisations by imposing the policy that *an organisation cannot allocate the same resource to two conflicting goals*. We can see this policy as a more general version of Brewer and Nash's Chinese Wall policy [5], imposed on the resources used in a VO.

For VO management, we abstract from the traditional secure VO lifecycle [3] by modelling three main phases:

- *Selection*: for each VO goal, this phase selects from the VBE the set of organisations that can achieve that goal; each organisation allocates the resources it needs to achieve the goal. For the purpose of this paper, this is the main phase of interest, where our Chinese Wall policy is enforced.
- *Operation*: for each VO goal, this phase represents the interaction (collaboration) among participant organisations in order to achieve that goal.
- *Dissolution*: when all goals in the VO have been achieved, the VO finalises its execution and dissolves.

The model of VOs we present here is motivated by several examples. Among these is the management information concerning scientific experiments undertaken using large facilities. In this environment, a scientist from one organisation may be working in collaboration with other organisations on one experiment, whilst his colleagues from the same organisation may be working on another experiment in a different collaboration on the same facility. The facility providers themselves, in particular, are often collaborators in many of the experiments.

In many cases, there is a need to ensure the results of one experiment are kept confidential from another over some period of time in order to ensure the correct attribution of credit in the publication of the results. In a few cases, even the "meta information", that a particular scientist is working on a particular experiment may be considered confidential as it may disclose that a particular direction is being pursued in the research.

Here, experiments are seen as goals, the information as a resource, and the collaborators as the organisations in a VO. Similar issues arise in modelling collaborations using other scientific resources such as the computing resources provided by the UK National Grid Service (http://www.grid-support.ac.uk/).

3 Background on Event-B

Event-B [2] is a refinement methodology that is an extension of the B language [1]. The refinement methodology can be used by software architects to incrementally develop a model of a system starting from the initial, most abstract, model sometimes called the specification and following gradually onto further layers of detail until a model with satisfactory detail is reached. If successful, the most detailed model will be the implementation itself.

Each layer has preconditions and postconditions associated with it in the style of Hoare logic [12] and refinement ensures the consistency of the development process by strengthening the postconditions and/or the preconditions associated to each layer thus removing nondeterminism until a deterministic implementation is reached. In fact, invariants denoting desirable behaviour can be specified at the level of each layer as well as across different layers, known as *gluing invariants*. Refinement then allows us to verify that these invariants are true.

In Event-B, a system is modeled as a *machine* which is composed of a local state in the form of *variables* and any number of *events*. Events consist of the following elements: a name, *Event*, any number of guards, P, and any number of generalised substitutions, T, as follows:

$$\mathit{Event} \stackrel{\mathtt{def}}{=} \textbf{WHEN } P \textbf{ THEN } T \textbf{ END}$$

The syntax of generalised substitutions is defined in Figure 1.

The **skip** is a do-nothing substitution, which does not affect the machine's state. The deterministic substitution, $x := E$, assigns to a variable, x, the value of an expression, E. Finally, in a non-deterministic substitution, it is possible to choose

skip	Do nothing
$x := E$	Deterministic substitution
ANY x **WHERE** P **THEN** T **END**	Non-deterministic substitution

Fig. 1. The syntax of generalised substitutions.

non-deterministically local variables, x, that will render the guard P true. If this is the case, then parallel substitutions, T, can be applied. Otherwise, nothing happens. We sometimes write the syntactic sugar, $x :\in \mathit{Set}$ as a short form of **ANY** z **WHERE** $z \in \mathit{Set}$ **THEN** $x := z$ **END**. An event may be fired (i.e. its substitutions applied) as soon as its conditional guards are satisfied. If more than one event is ready to fire, then one is picked non-deterministically.

Machines have *contexts* that they can see. A context has the following elements: a name, a set of carrier sets, a set of constants and a set of axioms. Carrier sets are essentially user-defined types and constants must have a particular type (primitive or user-defined). Usually, axioms are used to express such constant types and any other truths about the context elements.

For a comprehensive description of the Event-B language and its formal meaning, we refer the reader to more detailed references such as [14].

4 Isolation of Conflicting Goals

The first Event-B model we present in this paper is an abstract model of VOs that defines a VO as a set of goals with organisations collaborating to achieve those goals. The model highlights potential conflict-of-interest situations where a member organisation may pursue two or more conflicting goals during its VO history. The main components of the model are a context called VBE as shown in Figure 2, which represents a VBE, and a machine called VO illustrated in Figure 3, which represents the VO lifecycle.

CONTEXT VBE

SETS
Goals, Status, Organisations

CONSTANTS
Sl, Op, Stop, CoI

AXIOMS
Status = {Sl, Op, Stop}, Sl $\neq$ Op, Sl $\neq$ Stop, Op $\neq$ Stop,
Goals $\neq \emptyset$, Organisations $\neq \emptyset$,
CoI $\subseteq \mathbb{P}$(Goals), $\forall$e.e$\in$CoI $\Rightarrow$ card(e)=2, $\forall$g.g$\in$Goals $\Rightarrow$ {g,g}$\notin$CoI

END

Fig. 2. The abstract context VBE.

The VBE context defines three types: The non-empty type Goals, which represents the set of possible goals in some VBE; the non-empty type Organisations, which represents the set of VBE organisations willing to participate in potential VOs; and finally the type Status, which contains the flags Sl for the Selection phase, Op for the Operation and Dissolution phases and Stop, which marks the end of the VO lifecycle (when no further events are possible). The context also defines the set, CoI, which denotes the global conflict of interest among goals in the VBE. The definition of CoI states that it is a powerset of all two-element sets representing any two goals that are in conflict with each other. It also states that a goal cannot be in conflict with itself.

The VO machine captures essentially three phases of the VO lifecycle: Selection, Operation and Dissolution. The machine defines a number of local variables including the set of goals it is pursuing, a subset of those goals that the VO has completed and the current goal it is working on. The machine also defines a collaboration model variable, collM, which is a relation from one of its uncompleted goals (the current goal) to a set of organisations that will collaborate to achieve that goal. In addition to the current collaboration model, there is also the collaboration history, collH, which is a relation from any completed goal to the organisations that have collaborated in the past to achieve that goal. The machine also defines a status flag, which indicates the next event to be fired.

An interesting variable in the machine is the conflict of interest relation, coi, which is a relation from any organisation to a pair of goals denoting the fact that the two goals are conflicting and that they must be maintained isolated internally within the organisation throughout its history of collaborations in the VO. For example, if an organisation, org, collaborates on a goal, g1, and then later collaborates on another goal, g2, such that $\{g1,g2\} \in CoI$, then coi will record the element $(org \mapsto (g1 \mapsto g2))$ as one of its memebers.

The creation of the VO is modeled through a non-deterministic initialisation event in which the set of VO goals is chosen from the overall set of VBE goals. We next describe the rest of the events in the machine as follows:

- The Selection event: This is the first event that is executed after the initialisation event and in it, an uncompleted goal, aGoal, is chosen and nominated as the current goal on which the VO is working along with a suitably typed collaboration model, aCollM, which must contain at least one organisation that will work on aGoal. The event also selects a suitably typed coi relation, called acoi, which maps every organisation in the range of aCollM to a pair whose first element is aGoal and the second element is a conflicting goal on which the organisation worked on in the past (in collH). Once these conditions are satisfied, Selection performs a number of substitutions to update the currentGoal, collM, coi and status variables. The status variable is set to Op to indicate that the machine is ready to enter the Operation event.

  It is interesting to note here that the coi relation is *not* intended to be a condition for selecting collaborations but rather an *indication* to the requirement of the isolation of conflicting goals within organisations.

```
MACHINE VO SEES VBE

VARIABLES
status, goals, completedGoals, currentGoal, collM, collH, coi

INVARIANTS
status ∈ Status ∧ goals ∈ ℙ1(Goals) ∧ completedGoals ⊆ goals ∧ currentGoal ⊆ goals
collM ∈ currentGoal ↔ Organisations ∧ collH ∈ completedGoals ↔ Organisations ∧
coi ∈ Organisations ↔ (goals × goals) ∧

/* The Conflict of Interest Invariant */
(∀ org,g. org ∈ ran(collM) ∧ {g}=currentGoal ⇒
  (∀g1.g1∈completedGoals ∧ (g1↦org)∈collH ∧ {g,g1}∈CoI ⇒ (org↦(g↦g1))∈coi)) ∧
(∀ g0,g1,org. (g0↦org) ∈ collH ∧ (g1↦org) ∈ collH ∧ {g0,g1} ∈ CoI ⇒ (org↦(g0↦g1)) ∈ coi)

INITIALISATION
BEGIN
goals :∈ ℙ1(Goals) ‖ completedGoals := ∅ ‖ currentGoal := ∅ ‖
collM := ∅ ‖ collH := ∅ ‖ coi := ∅ ‖ status := Sl
END

EVENT Selection
ANY
aGoal, aCollM, acoi
WHERE
status = Sl ∧ aGoal∈(goals\completedGoals) ∧
aCollM∈{aGoal} ↔ Organisations ∧ ran(aCollM)≠∅ ∧
acoi∈ran(aCollM)↔({aGoal}× completedGoals) ∧

/*The Chinese Wall Guard*/
∀org.org∈ran(aCollM) ⇒
   (∀g1.g1∈completedGoals∧(g1↦org)∈collH∧{aGoal,g1}∈CoI ⇒ (org↦(aGoal↦g1))∈acoi)  ∧
/*The Minimality of acoi Guard*/
∀org.org∈ran(aCollM) ⇒
   (∀g1.¬g1∈completedGoals∨¬(g1↦org)∈collH∨¬{aGoal,g1}∈CoI ⇒ ¬(org↦(aGoal↦g1))∈acoi)
THEN
currentGoal := {aGoal} ‖  collM := aCollM ‖ coi := coi ∪ acoi ‖ status := Op
END

EVENT Operation
WHEN
status = Op
THEN
completedGoals := completedGoals ∪ currentGoal ‖ collH := collH ∪ collM ‖ status := Sl
END

EVENT Dissolution
WHEN
status = Sl ∧ completedGoals = goals
THEN
status := Stop
END

END
```

Fig. 3. The VO abstract machine.

- The Operation event: In this event, the VO adds to the set of completed goals the current VO goal and at the same time, updates the collaboration history, collH, to include the current collaboration model, collM. The status flag is reset to the Sl value to indicate that the VO is now ready to select a new goal.
- The Dissolution event: Once the set of completed goals has reached the set of VO goals, the VO will have arrived at the end of its lifetime and it is now ready to dissolve. This is modeled as setting the status flag to Stop, at which point the VO cannot select any event.

The machine defines the following conflict of interest invariant.

Invariant 1 (Conflict of Interest Invariant)
For any organisation, org, a current goal, g, and past completed goals, g0 and g1, then the following holds true:

- $org \in ran(collM) \wedge g \in currentGoal \Rightarrow (\forall g1.g1 \in completedGoals \wedge (g1 \mapsto org) \in collH \wedge \{g,g1\} \in CoI \Rightarrow (org \mapsto (g \mapsto g1)) \in coi)$, *and*
- $(g0 \mapsto org) \in collH \wedge (g1 \mapsto org) \in collH \wedge \{g0,g1\} \in CoI \Rightarrow (org \mapsto (g0 \mapsto g1)) \in coi$

Proof. We give here a proof sketch. To prove the invariant, we must show that the substitutions in all of the events (Initialisation, Selection, Operation and Dissolution) respect it. This is easy to show for the cases of Initialisation and Dissolution. However, for the case of Selection, we have to prove that the acoi local variable preserves the invariant, since acoi is added to coi. This can be shown using the Chinese Walls guard and the minimality guard, which are used in selecting the acoi relation. These guards ensure that acoi contains only the right tuples and nothing else. For the second part of the invariant, Selection only adds acoi to coi, so the history recorded in coi is still preserved by coi ∪ acoi. For the case of Operation, the first part of the invariant can be easily proven to hold for the new values of completedGoals and collH variables. For the case of the second part, we must prove that the new values of collH (which include the current collM) preserve the second part of the invariant. This can be done by showing that the domain of collM is a singleton and that no goal is conflicting with itself. □

The first implication of the invariant states that if an organisation is collaborating towards some current goal and has in the past collaborated on some conflicting goal, then the conflict of interest relation coi must register this fact. On the other hand, the second implication states that if an organisation has collaborated in the past on two conflicting goals, then that fact will have been captured by the coi relation. This invariant is essentially maintaining information relating to the need for a Chinese Wall separation between conflicting goals that an organisation is (has been) working on.

5 Separation of Resources

To enforce the isolation of conflicting goals captured in the abstract model of the previous section, we introduce more detail into the model. This detail is represented by the concept of *resources* committed by organisations owning them to the cause of achieving the goals of a VO. The main idea driving the concrete model presented in this section is to enforce a separation, within each organisation, among the resources allocated to conflicting goals. Hence an organisation wishing to work on conflicting goals must not use the same resource for both goals.

The concrete model is composed from a refined context, called VBEResources, as shown in Figure 4, and a refined machine, called VOResources, as shown in Figure 5, where, for the sake of conciseness, we have only included the extra detail. The VBEResources context introduces a new non-empty type, Resources, to represent all the resources advertised by organisations in the VBE, and a function, ownedBy, which denotes the ownership of resources by organisations. The fact that ownedBy is a function and not a relation implies that every resource is owned by a single organisation.

CONTEXT VBEResources **REFINES** VBE

SETS
Resources

CONSTANTS
ownedBy

AXIOMS
Resources $\neq \emptyset$, ownedBy $\in$ Resources $\rightarrow$ Organisations

END

Fig. 4. The refined context VBEResources.

On the other hand, the refined machine introduces two new variables: the allocation model, allocM, and the allocation history, allocH. The former is a relation from the current goal to resources allocated to it and the latter is a relation from completed goals to resources allocated to them in the past. The machine is still composed from the same events representing the VO lifecycle, namely, Selection, Operation and Dissolution.

In the Selection event, a local variable, anAllocM, represnting an allocation model, is chosen that will achieve the current working goal, aGoal, using a non-empty set of resources. This variable is then assigned to allocM. The allocation model variable must sound and complete with respect to the collaboration model variable introduced in the abstract machine. This is ensured by the following two conditions, respectively:

$$\forall r.(aGoal \mapsto r) \in anAllocM \Rightarrow (aGoal \mapsto ownedBy(r)) \in aCollM$$

$$\forall o.(aGoal \mapsto o) \in aCollM \Rightarrow (\exists r.ownedBy(r) = o \wedge (aGoal \mapsto r) \in anAllocM)$$

The soundness condition ensures that allocated resources belong to VO members collaborating on the current goal and therefore there are no foreign resources. The completeness condition, on the other hand, ensures that every collaborating organisation commits at least one resource to the achievement of the goal it is pursuing,

MACHINE VOResources **REFINES** VO **SEES** VBEResources

VARIABLES
$\ldots$ allocM, allocH $\ldots$

INVARIANTS
allocM $\in$ currentGoal $\leftrightarrow$ Resources $\wedge$
allocH $\in$ completedGoals $\leftrightarrow$ Resources $\wedge$

/*The Chinese Wall Invariant*/
($\forall$r,g0,g1.g0$\in$currentGoal $\wedge$ (g1$\mapsto$r) $\in$ allocH $\wedge$ $\neg$(g1$\mapsto$r)$\in$allocM $\wedge$ (ownedBy(r)$\mapsto$(g0$\mapsto$g1))$\in$coi $\Rightarrow$
$\neg$(g0$\mapsto$r)$\in$allocM) $\wedge$
($\forall$r,g0,g1.(g0$\mapsto$r) $\in$ allocH $\wedge$ (g1$\mapsto$r) $\in$ allocH $\Rightarrow$ $\neg$(ownedBy(r)$\mapsto$(g0$\mapsto$g1))$\in$coi) $\wedge$
$\ldots$

INITIALISATION
BEGIN
$\ldots$ allocM := $\emptyset$ $\|$ allocH := $\emptyset$ $\ldots$
END

EVENT Selection
ANY
$\ldots$ anAllocM $\ldots$
WHERE
anAllocM$\in$\{aGoal\}$\leftrightarrow$Resources $\wedge$ ran(anAllocM)$\notin\emptyset$ $\wedge$

/*Soundness of anAllocM*/
$\forall$r.(aGoal$\mapsto$r)$\in$anAllocM $\Rightarrow$ (aGoal$\mapsto$ownedBy(r))$\in$aCollM $\wedge$

/*Completeness of anAllocM*/
$\forall$o.(aGoal$\mapsto$o)$\in$aCollM $\Rightarrow$ ($\exists$r.ownedBy(r)=o $\wedge$ (aGoal$\mapsto$r)$\in$anAllocM) $\wedge$

/*Chinese Wall Guard*/
$\forall$r,g.(g$\mapsto$r)$\in$allocH $\wedge$ $\neg$(g$\mapsto$r)$\in$anAllocM $\wedge$ (ownedBy(r)$\mapsto$(aGoal$\mapsto$g))$\in$ coi $\cup$ acoi $\Rightarrow$
$\neg$(aGoal$\mapsto$r)$\in$anAllocM
$\ldots$
THEN
$\ldots$ allocM := anAllocM $\ldots$
END

EVENT Operation
WHEN
$\ldots$
THEN
$\ldots$ allocH := allocH $\cup$ allocM $\ldots$
END

EVENT Dissolution
$\ldots$
END

END

Fig. 5. The VOResources concrete machine.

i.e. there are no idle organisations.

The Chinese Wall guard ensures that the allocation model variable is chosen such that no resource is allocated to the current working goal that was allocated, in the past, to a conflicting goal:

$$\forall r, g.(g \mapsto r) \in allocH \wedge \neg(g \mapsto r) \in anAllocM \wedge (ownedBy(r) \mapsto (aGoal \mapsto g)) \in coi \cup acoi \quad \Rightarrow \quad \neg(aGoal \mapsto r) \in anAllocM$$

The next event, Operation, among other actions updates the allocation history, allocH, by adding to it the current value of the allocation model, allocM.

The following gluing invariant provides a link between coi and the resource allocation model and history relations.

Invariant 2 (The Chinese Walls Invariant)
For any resource, r, and two goals, g0 and g1, then the following holds true for all the refined events:

- $\forall r,g0,g1.g0 \in currentGoal \wedge (g1 \mapsto r) \in allocH \wedge \neg(g1 \mapsto r) \in allocM \wedge (ownedBy(r) \mapsto (g0 \mapsto g1)) \in coi \Rightarrow \neg(g0 \mapsto r) \in allocM$
- $\forall r,g0,g1.(g0 \mapsto r) \in allocH \wedge (g1 \mapsto r) \in allocH \Rightarrow \neg(ownedBy(r) \mapsto (g0 \mapsto g1)) \in coi$

Proof. The proof of the invariant relies on showing that substitutions in every event in the concrete machine preserve the invariant. This can be shown using the Chinese Wall guard for the Selection event. In the Operation event, the proof is based on showing that the domain of allocM will always be a singleton goal and that no goal is conflicting with itself. □

This invariant states that the current goal will never share a resource within an organisation that was allocated in the past to a conflicting goal within the same organisation. This isolation of resources will be based on information recorded by the coi relation in the abstract machine. The invariant demonstrates how the Chinese Wall policy is refined from the abstract level to the concrete level.

6 Related Work

Conflicts of interest have been a topic of interest in information security since the early days when Brewer and Nash proposed the Chinese Wall security policy [5]. We have presented here a more general policy for managing conflicts of interest, which indeed it has been inspired by broad versions on the Chinese Wall policy. In Kelley Sobel and Alves-Foss' Chinese Wall model [16], after an individual has accessed an object, s/he is not permitted to access data from an object that is classified as having a conflict-of-interest. In our model, after a VO resource is allocated to a goal, it is not then permitted to allocate such resource to conflicting goals.

We have followed the B-method approach to security, in which security properties are represented in terms of invariants that are preserved by a process of step-wise refinement [4]. This technique has been successfully applied in modeling security properties of network monitors [17] and the Mondex Electronic Purse [7].

Conflicts of interest arise naturally in dynamic coalitions such as virtual organisations. To our knowledge, there is not previous work on analysing conflicts of interest in virtual organisations. Recently, Bryans *et al* [6] have modeled formally general aspects of dynamic coalitions paying especial attention to information flow. They use the Vienna Development Method (VDM) [13] to construct a suit of models representing important aspects of coalitions: membership policies, information discovery, and information transfer. However, the conflict-of-interest dimension is not included in their analysis. In [18], Zhou and Foley describe a logic-based language that provides a foundation for coalition regulation and security policies. They propose a formal framework for regulating the establishment of dynamic coalitions in which coalitions are formed with the involvement of founders, constructors and oversight, and do not rely on the traditional notion of a super-administrator; their emphasis is on coalition delegation.

7 Conclusion

We have modeled conflicts of interest in VOs using the Event-B specification language [2] supported by the RODIN toolkit (`http://rodin.cs.ncl.ac.uk/`). The main elements in our abstract specification (Section 4) are goals and organisations; events model the main phase of a VO lifecycle: goal selection, operation, and dissolution. The security property is represented by marking – i.e. adding to the coi relation — those organisations that participate in conflicting goals. The main elements in the concrete specification (Section 5) are goals, organisations and resources. The security property states that an organisation cannot allocate the same resource to two conflicting goals. The nature of the refinement that we verified is safety refinement, that is, any behaviour (trace of events) of the concrete model must be behaviour of the abstract one.

One advantage of using proof support tools like RODIN was that we were able to evolve the model gradually, e.g. by redefining the different elements and relations of the model till the most suitable ones were reached. This eventually helped clarify our understanding of the domain of the problem we are dealing with.

This work being part of project GridTrust (`http://www.gridtrust.eu`), we aim in the short term to apply the same approach to analyse conflicts of interest in the case of inter-enterprise knowledge management systems, since this is one of the applications being developed in the GridTrust project. In the medium term, we plan to use Event-B to model other security properties in VOs, in particular role-based access control [10] and continuous usage of resources [15]. We also plan to investigate further layers of detail in the model beyond the resource layer. In particular, we are interested in adding datasets to the model in which case it will be possible to express invariants about information flow.

Acknowledgment

The work reported here was partially funded by the IST FP6 GridTrust project (http://www.gridtrust.eu), contract No. 033827.

References

[1] J. R. Abrial. *The B Book*. Cambridge University Press, 1996.

[2] J. R. Abrial and L. Mussat. Introducing Dynamic Constraints in B. In D. Bert, editor, *B98: Recent Advances in the Development and Use of the B Method*, volume 1393 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[3] A. E. Arenas, I. Djordjevic, T. Dimitrakos, L. Titkov, J. Claessens, C. Geuer-Pollmann, E. C. Lupu, N. Tiptuk, S. Wesner, and L. Schubert. Toward Web Services Profiles for Trust and Security in Virtual Organisations. In *Collaborative Networks and their Breeding Environments (PRO-VE 2005)*. Springer, 2005.

[4] P. Bieber and N. Boulahia-Cuppens. Formal Development of Authentication Protocols. In *BCS-FACS Sixth Refinement Workshop*, 1994.

[5] D. Brewer and M. Nash. The Chinese Wall Policy. In *IEEE Symposium on Research in Security and Privacy*. IEEE, 1989.

[6] J. W. Bryans, J. S. Fitzgerald, C. B. Jones, and I. Mozolevsky. Formal Modelling of Dynamic Coalitions with an Application in Chemical Engineering. In *2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*. IEEE, 2006.

[7] M. Butler and D. Yadav. An Incremental Development of the Mondex System in Event-B. *Submitted to Formal Aspects of Computing*, 2007.

[8] L. M. Camarihna-Matos and H. Afsarmanesh. Elements of a Base VE Infrastructure. *Journal of Computers in Industry*, 51(2):139–163, 2003.

[9] L. M. Camarihna-Matos and H. Afsarmanesh, editors. *Collaborative Networked Organisations — A Research Agenda for Emerging Business Models*. Kluwer, 2004.

[10] D. Ferraiolo and D.R. Kuhn. Role-Based Access Controls. In *15th Annual Conference on National Computer Security*, 1992.

[11] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[12] C.A.R. Hoare. An Axiomatic Basis for Computer Programming. *Communications of the ACM*, 12(10):576–585, October 1969.

[13] C. B. Jones. *Systematic Software Development using VDM*. Prentice Hall International, 1990.

[14] C. Métayer, J. R. Abrial, and L. Voisin. Event-B Language. Rodin Deliverable D3.2, 2005.

[15] J. Park and R.S. Sandhu. The $\mathrm{UCON}_{\mathrm{abc}}$ Usage Control Model. *ACM Transactions on Information and System Security*, 7(1):128–174, February 2004.

[16] A. E. Kelley Sobel and J. Alves-Foss. A Trace-Based Model of the Chinese Wall Security Policy. In *Proceedings of the National Information System Secuirity Conference*, 1999.

[17] N. Stouls and M-L. Potet. Security Policy Enforcement through Refinement Process. In *B2007: The 7th International B Conference*, Lecture Notes in Computer Science. Springer-Verlag, 2007.

[18] H. Zhou and S. N. Foley. A Framework for Establishing Decentralized Secure Coalitions. In *Proceedings of IEEE Computer Security Foundations Workshop*. IEEE, 2006.