On Formalising Policy Refinement in Grid Virtual Organisations

Benjamin Aziz

Abstract Grid computing is a global-computing paradigm focusing on the effective sharing and coordination of heterogeneous services and resources in dynamic, multi-institutional Virtual Organisations (VOs). This paper presents a formal model of VOs using the Event-B specification language. We have followed a refinement approach to develop goal-oriented VOs by incrementally adding their main elements: goals, organisations and services. Our main interest is in the problem of policy refinement in VOs, so policies are represented as invariants that should be maintained throughout the refinement process. As an illustration, we show how a VO resource-usage policy is represented at different levels of abstraction.

1 Introduction

Grid computing is a global-computing paradigm focusing on the effective sharing and coordination of heterogeneous services and resources in dynamic, multi-institutional Virtual Organisations (VOs) [13]. A Grid VO can be seen as a temporary or permanent coalition of geographically dispersed organisations that pool services and resources in order to achieve common goals. This paper presents a formal model of VOs using the Event-B specification language [3]. We have followed a refinement approach to develop goal-oriented VOs by incrementally adding their main elements: goals, organisations and services. Our main interest is in the problem of policy refinement in VOs.

Policy refinement is the process of transforming a high-level abstract policy specification into a low-level concrete one [16]. Current approaches to policy refinement in distributed and dynamic systems suppose that the refinement of the abstract system entities into the concrete objects/devices is done as a previous phase to the refinement of policies, by assuming there exist pre-defined hierarchies of concrete

Benjamin Aziz
University of Portsmouth, Portsmouth, United Kingdom, e-mail: benjamin.aziz@port.ac.uk

objects/devices [22] or by taking the concrete system architecture as an input [8]. Here, we use the stepwise refinement approach [6] to develop simultaneously both the system entities and their policies. In our case, policies are represented as invariants that should be maintained throughout the refinement process. We illustrate this approach by analysing the case of a resource-usage policy, the so-called *cost-balancing policy*, where the cost of achieving a goal in a VO is divided equally among the VO members. This is a particular case of the $1/N$ policy [23], a representative Grid policy indicating that all resource utilisation is equally distributed among the VO-member resources.

The work presented here has a twofold aim; on one hand, we would like to gain a more formal understanding of VOs and their lifecycle, especially in the presence of policy contraints. On the other hand, we would like to experiment with the process of designing VOs following the refinement process paying particular attention to non-functional properties such as resource usage and security. In recent years, the need for adopting rigorous approaches for designing distributed systems such as VOs has risen due to the various challenges posed by the use of such systems in safety and security critical collaborative environments such as collaborative engineering in the aerospace domain [14], Grid-based operating systems [17] and others.

The structure of the paper is the following. Section 2 introduces the main elements of a VO and the VO life cycle. Next, Section 3 gives a brief overview of Event-B. Section 4 presents a motivating case scenario involving cost-balancing policies. Section 5 presents our abstract model of VOs; a model containing only goals and representing the VO lifecycle. An intermediate refinement is described in Section 6, which includes goals and organisations. Our concrete model is presented in Section 7, including goals, organisations and services. Section 8 presents related work and finally, Section 9 concludes the paper and highlights future work.

2 On Virtual Organisations and Their Lifecycle

The entities that form a VO are drawn from a "club of potential collaborators" called a *Virtual Breeding Environment* (VBE) [10]. A VBE can be defined as an association of organisations subscribing to a base long term cooperation agreement, adopting common operating principles and infrastructure with the objective of participating in future potential VOs. In this paper, we take 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 the services provided to fulfill such goals.

For the management of a VO, we are following a VO life-cycle adopted by other projects such as ECOLEAD [11] and TrustCoM [5]. The life-cycle includes the following phases:

- VO Identification: In this phase, the VO Administrator sets up the VO by selecting potential partners from the VBE, using search engines or registries. In our model, we will be looking for partners that can achieve the goals identified in the

VO. The identification phase ends with a list of candidates that potentially could perform the goals needed for the current VO.

- VO Formation: In the formation phase, the initial set of candidates is reduced to a set of VO members. This process may involve a negotiation between potential partners. After this has been completed, the VO is configured and can be considered to be ready to enter the operation phase.
- VO Operation: The operation phase could be considered the main life-cycle phase of a VO. During this phase the VO members contribute to the VOs task(s) by executing pre-defined business processes (e.g. service orchestration) to achieve the VO goals. Membership and structure of VOs may evolve over time in response to changes of objectives or to adapt to new opportunities in the business environment; this is a feature we are not considering in the current version of our model.
- VO Dissolution: During dissolution, the VO structure is dissolved and final operations are performed to annul all contractual binding of the partners.

Figure 1 illustrates the VO lifecycle. As part of our model, we show in the paper a formalisation of the VO lifecycle, where each VO phase is modelled as an event. In our view, a VO policy is a property that should be respected across the VO phases. We model the initial actions needed to enable the integration of organisations into a VO as an additional phase called Initialisation.

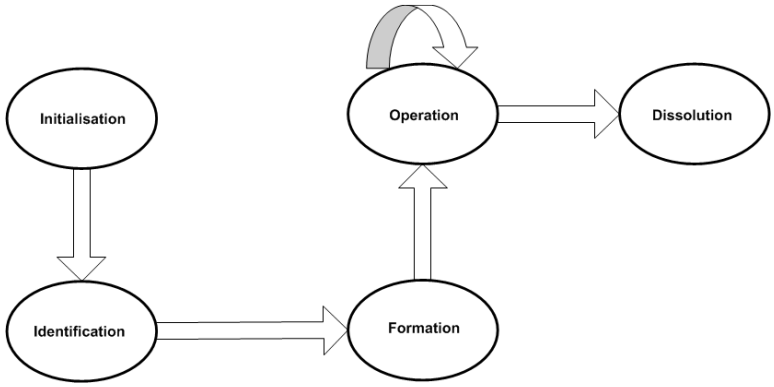

Fig. 1 The VO Lifecycle.

3 A Brief Overview of Event-B

Event-B [2] is an extension of Abrial's B method [1] for modelling distributed systems. This section presents a brief overview of Event-B; we refer the reader to [2, 12] for a more complete description of this formal method. Modularity is central to the

Event-B method and this is achieved by structuring specifications and development into *Machines*. A *machine* encapsulates a local state and provides operations on the state, as shown in Figure 2.

	MACHINE AM
	SEES AC
CONTEXT AC	**VARIABLES** v
SETS T	**INVARIANT** I
CONSTANTS c	**INITIALISATION** *Init*
AXIOMS A	**EVENTS**
END	E_1 = **WHEN** G **THEN** S **END**
	$\cdots$
	$E_n = \cdots$
	END

Fig. 2 Abstract Machine Notation in Event-B.

The **CONTEXT** component specifies the types and constants that can be used by a machine. It is uniquely identified by its name AC and includes clauses **SETS**, defining carrier sets (types); **CONSTANTS**, declaring constants; and **AXIOMS**, defining some restrictions for the sets and including typing constraints for the constants in the way of set membership.

A machine is introduced by the **MACHINE** component, which is uniquely identified by its name AM. A machine may reference a context, represented by clause **SEES**, indicating that all carrier sets and constants defined in the context can be used by the machine. Clause **VARIABLES** represents the variables (state) of the model, which are initialised in *Init* as defined in the **INITIALISATION** clause. The **INVARIANT** clause describes the invariant properties of the variables, denoting usually typing information and general properties. These properties shall remain true in the whole model and in further refinements. The **EVENTS** clause defines all the events (operations) describing the behaviour of the system. Each event is composed of a guard G (a predicate) and an action S, which is a statement, such that if G is enabled, then S can be executed. If several guards are enabled at the same time then the triggered event is chosen in a nondeterministic way.

Statements in the bodies of events have the following syntax:

$$
\begin{array}{l}
S == x := e \ \mid \\
\quad \textbf{IF } cond \textbf{ THEN } S1 \textbf{ ELSE } S2 \textbf{ END} \ \mid \\
\quad x :\in T \ \mid \\
\quad \textbf{ANY } z \textbf{ WHERE } P \textbf{ THEN } S \textbf{ END} \ \mid \\
\quad S1 \parallel S2
\end{array}
$$

Assignment and conditional statements have the standard meaning. The non-deterministic assignment $x :\in T$ assigns to variable x an arbitrary value from the given set (type) T. The non-deterministic block **ANY** z **WHERE** P **THEN** S **END** introduces the new local variable z that is initialised non-deterministically according to the predicate P and then used in statement S. Finally, $S1 \parallel S2$ models parallel

(simultaneous) execution of $S1$ and $S2$ provided they do not have conflict on state variables. Statements are formally defined using a weakest precondition semantics.

In order to be able to ensure the correctness of a system, a machine should be consistent and feasible. This is assured by proving the initialisation is feasible and establishes the invariant, and then each event is feasible and preserves the invariant. Proof obligations are generated automatically and verified using the RODIN toolkit [20]. Proof obligations are generated via before-after predicates denoting the relation between the variable values before and after the execution of a statement.

Event-B supports stepwise refinement, the process of transforming an abstract, non-deterministic specification into a concrete, deterministic, system that preserves the functionality of the original specification. We use a particular refinement method, *superposition refinement* [7], where the state space is extended while preserving the old variables. During the refinement process, new features that are suggested by the requirements are represented by new variables added to the system. Simultaneously, events are refined to take the new features into account. This is performed by strengthening their guards and adding substitutions on the new variables.

3.1 Our Approach

The general approach we adopt in this paper involves the following steps:

- First, we use Event-B to model, at an abstractb level, a specific system. This will be in our case the system of goal-oriented VOs.
- Second, we use the refinement mechanism supported by Event-B to add more detail gradually to the original abstract model, until one arrives at the required level of detail. In this case, this will be realised by refining our abstract goal-oriented VOs to VOs with organisations and goal costs, then again refine further to VOs with service sets.
- Finally, we express any policy constraints we need (in our case, the cost balancing constraint we discuss in the next section) in terms of the machine invariants starting from some level of detail in the refinement chain. This could either start at the abstract level, or at any level of the refined machines. We then show that the same policy (invariant) is respected and upheld by the lower levels of refinement.

This approach is general and can be applied to any domain and with any policy requirements. The rest of the paper considers only one example of the application of this approach.

4 Case Study: $1/N$ Cost-Balancing Policy in Auction-based Routing VOs

The case study that motivated this paper is based on an auctioning VO that allows transportation customers in a supply chain scenario to place their requests for transport on an online auctioning system. Transportation companies can then bid for these requests through a transporters' portal at the backend of the auctioning system. The collection of the customers and the service providers forms one VO called the *Auctioning VO*.

At the same time, each transportation company can form a second VO called a *Routing VO*, which will involve along with the transportation company all the necessary computational resources needed for computing the routing calculation resulting in the bid offer. The highly complex computations could be outsourced to other organisations, which is why the Routing VO is needed. In both VOs the manager is the Transporter Association Portal (TAPortal), through which the administrator creates and populates the two VOs. This scenario is depicted in Figure 3.

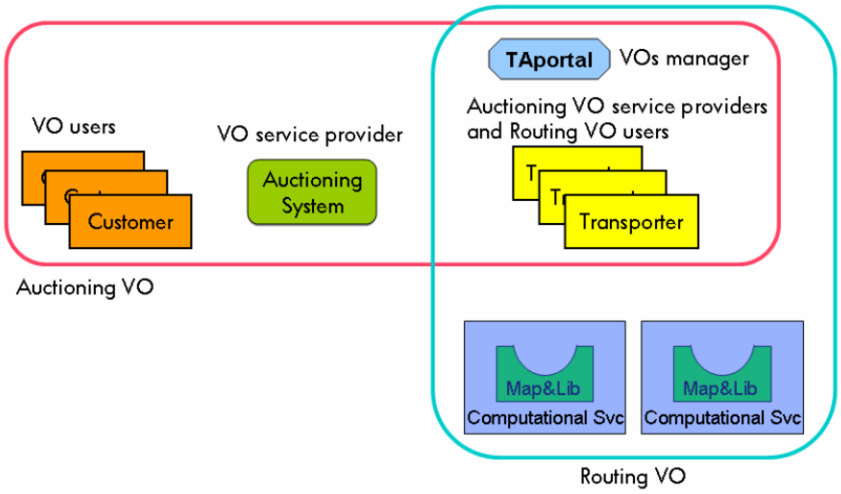

Fig. 3 Auctioning-based Routing VOs.

In the above case study, a *cost balancing* policy would be desireable in the Auctioning VO, in the event that a customer of the VO is planning to divide their transportation task among N number of service providers, while determning what the cost associated with each transportation stage (service) would be. The bid calculated by each transporter is then compared to the budget advertised by the customer in their request, and the winning bid is the one with the best cost estimate.

Such a policy is known as a $1/N$ *cost-balancing* policy, and it is one example of VO-wide policies that are typical in Grid systems [23], which deal with the prob-

lem of managing VO resources by dividing equally the resource utilisation among the member organisations. Such policies are useful in critical applications [14] and Grid-based operating systems [17], since they facilitate the regulation of resource usage.

Informally, our version of the policy states that the cost of achieving a goal is divided equally among the VO members (organisations) that are collaborating toward achieving that goal. This then implies that the cost of services employed by each organisation toward the goal will be equal to the cost of services employed by any of its sibling organisations. Ideally, this cost must not exceed the budget allocated to the organisation. Figure 4 illustrates this policy across the two layers of abstraction (organisations and services).

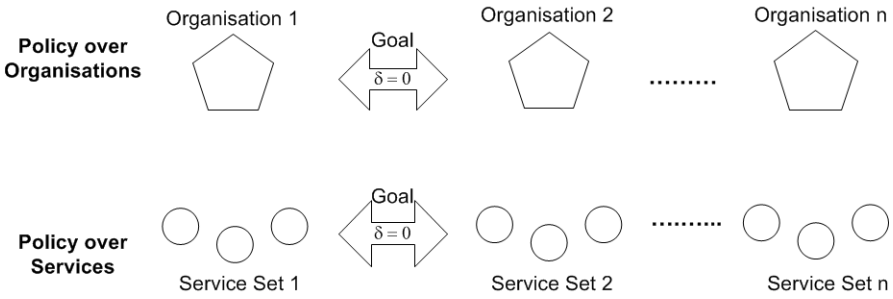

Fig. 4 The $1/N$ cost-balancing policy.

The policy is formalised in terms of a cost distance variable, $\delta \in \mathbb{N}$, which measures the difference between any two entities (organisations or sets of services) working on the same goal. When delta returns zero, then the policy becomes a $1/N$ cost-balancing policy, where N is the cardinality of the set of entities sharing the cost. On the other hand, if δ is set to some non-zero value, then this will imply that any two organisations are allowed to have some difference in their cost associated with achieving the main goal of the VO. It is outside the scope of this paper to determine what the value of δ should be, this will be largely dependant on each specific case of the auctioning problem.

The top layer in Figure 4 shows this policy ($\delta = 0$ for some Goal) among the various Organisations $1 \ldots n$, whereas in the lower more refined layer, we see the same policy this time on Service Sets $1 \ldots n$, where each set is the representation (refinement) of its corresponding organisation.

5 An Abstract Model of Goal-Oriented VOs

The first model of a VO is goal-oriented; it captures the idea that a VO is driven by the aim to achieve a set of goals that some VBE makes possible. The model defines

a machine, which represents the VO lifecycle as discussed in Section 2 based on this idea of goal-driven VOs. The machine and its context are shown in Figure 5.

MACHINE VO **SEES** VBE

VARIABLES
status, goals, completedGoals

INVARIANTS
/∗**Here we define the types of goals and completedGoals**∗/
status ∈ Status ∧ goals ∈ ℙ1(Goals) ∧ completedGoals ⊆ goals

INITIALISATION
goals :∈ ℙ1(Goals) ‖ completedGoals := ∅ ‖ delta := 0 ‖ status := Id
END

Identification
/∗**Nothing to identify**∗/
WHEN status = Id THEN status := Fr END

Formation
/∗**Nothing to form**∗/
WHEN status = Fr THEN status := Op END

Operation
/∗**Pick an uncompleted goal and achieve it**∗/
WHEN status = Op ∧ (completed_goals ≠ goals) THEN
ANY aGoal WHERE aGoal ∈ (goals\ completedGoals) THEN
completedGoals := completedGoals ∪ {aGoal} END

Dissolution
/∗**No more uncompleted goals, therefore stop**∗/
WHEN status = Op ∧ goals = completedGoals THEN status := Stop END

END

CONTEXT VBE

SETS
Goals, Status

CONSTANTS
Id, Fr ,Op, Stop

AXIOMS
Status = { Id, Fr, Op, Stop }
ℙ1(Goals) ≠ ∅ ∧ finite(Goals)

END

Fig. 5 The abstract machine, VO, and its abstract context, VBE.

The VBE is modelled as a context that introduces a carrier set (type) called Goals. Goals form a non-empty finite set. The context also includes the type Status, which is a flag representing the different phases of the VO lifecycle. The machine has four events corresponding to the four phases of the VO lifecycle as described in Section 2. The VO machine contains variables that represent the *status* (or VO lifecycle phase) of the machine, the *goals* of the VO and the *completed goals* of the VO. The machine is initialised such that the goals variable is assigned some non-empty value

from the VBE goals and so that the first event at which the machine commences is the Identification event.

The Identification event only changes the status flag to the next event (Formation). At this level of abstraction, there is no concept of organisations and therefore it is impossible to identify potential VO candidates. In the following event, Formation, again the only update to the machine's state is to change the status flag to indicate to the Operation event, also since there is no concept of organisations at this stage and hence, it is impossible to model VO membership formation. The Operation event is triggered as long as the set of completed goals has not yet reached the set of VO goals. When this is the case, a goal (aGoal) is chosen non-deterministically from the set of incomplete goals and added to the set of completed goals. Note that for simplicity, we do not model operational failure here. Finally, once the set of completed goals reaches the set of VO goals, the Dissolution event is triggered, which in turn sets the status flag to the Stop value indicating the end of the VO lifecycle.

This machine is too abstract to represent our cost-balancing policy, which refers to goals and organisations. Nevertheless, we have included it to show the modelling style we follow in the rest of the paper. The machine also demonstrates in an abstract manner that the aim of a VO lifecycle is to start and finish some specific goal.

6 Goal-Oriented VOs with Organisations

In the first refinement, we introduce the concepts of *organisations* and *goal cost*. The refined machine and its context are shown in Figure 6. The context VBERef1 is the refined VBE which introduces the type Organisations. The context also introduces two new constants, GoalCandidates and GoalCost. The former models the possible groups (sets) of organisations that when collaborating together can achieve a particular goal. The fact that GoalCandidates is a relation and not a function implies that there could be more than one such set of organisations per goal. The latter is a function that reflects the cost of achieving a goal as advertised by a set of organisations. Here we assume that cost is a stable value, which leads to GoalCost being a function rather than a relation.

The VORef1 machine consists again of the four VO lifecycle events; Identification, Formation, Operation and Dissolution. In the Identification event, the set of organisations that are candidates to join the VO are identified using the relation goalCandidates, which restricts the domain of GoalCandidates defined in the VBERef1 context to the set of VO goals. The next event is Formation, in which the VO members defined by the function, goalMembers, and their budget defined by the function, memberBudget, are updated. The goalMembers function is defined as being a functional subset of the more general goalCandidates relation. On the other hand, memberBudget is selected such that for an organisation operating towards achieving a goal, then the member budget assigned to that organisation is equal to the total cost of the goal divided by the cardinality of the set of organisations work-

MACHINE VORef1 **REFINES** VO **SEES** VBERef1

VARIABLES
status, goals, completedGoals, goalCandidates, goalMembers, delta, memberBudget

INVARIANTS
/∗**Type of goalCandidates**∗/
goalCandidates ∈ goals ↔ ℙ1(Organisations) ∧

/∗**Type of goalMembers**∗/
goalMembers ∈ goals → ℙ1(Organisations) ∧

/∗**Type of delta**∗/
delta ∈ ℕ

/∗**Type of memberBudget**∗/
memberBudget ⊆ Organisations → ℕ1 ∧

/∗**The 1/N cost-balancing policy invariant: VO members have equal budgets**∗/
$\forall g, o1, o2.g \in$ goals ∧ o1∈goalMembers(g) ∧ o2∈goalMembers(g) ⇒
(memberBudget($o1$) − memberBudget($o2$) = delta) ∨
(memberBudget($o1$) − memberBudget($o2$) = 0-delta)

INITIALISATION
goals :∈ ℙ1(Goals) ‖ completedGoals := ∅ ‖ goalCandidates := ∅ ‖ goalMembers := ∅
‖ delta := 0 ‖ memberBudget := ∅ ‖ status := Id END

Identification REFINES Identification
/∗**Identify potential candidates**∗/
WHEN status = Id THEN goalCandidates := goals ◁ GoalCandidates ‖ status := Fr END

Formation REFINES Formation
/∗**Form the VO organisation membership**∗/
ANY goalMembers0, memberBudget0 WHERE status = Fr ∧

/∗**The definition of goalMember0**∗/
goalMembers0 ∈ goals → ℙ1(Organisations) ∧ goalMembers0 ⊆ goalCandidates ∧

/∗**The definition of memberBudget0**∗/
memberBudget0 ∈ Organisations → ℕ1 ∧

/∗**The 1/N cost-balancing policy condition**∗/
$(\forall g, o.g \in goals \wedge o \in$ goalMembers0(g) ∧ card(goalMembers0(g)) ≠ 0 ∧ finite(goalMembers0(g)) ⇒
memberBudget0(o) = GoalCost($g \mapsto$ goalMembers0(g))÷card(goalMembers0(g)))

THEN goalMembers := goalMembers0 ‖ memberBudget := memberBudget0 ‖ status := Op END

Operation REFINES Operation
/∗**Operate on an uncompleted goal with the right member set**∗/
ANY aGoal, memberSet WHERE status = Op ∧ completedGoals ≠ goals ∧
aGoal ∈ (goals\ completedGoals) ∧ memberSet = goalMembers(aGoal) THEN
completedGoals := completedGoals ∪ {aGoal} END

Dissolution REFINES Dissolution
/∗**No more uncompleted goals therefore stop the VO lifecycle**∗/
WHEN status = Op ∧ goals = completedGoals THEN status := Stop END

END

CONTEXT VBERef1 **REFINES** VBE

SETS
Organisations

CONSTANTS
GoalCandidates, GoalCost

AXIOMS
GoalCandidates ∈ Goals ↔ ℙ1(Organisations) ∧ GoalCost ∈ Goals × ℙ1(Organisations) → ℕ1

END

Fig. 6 The first refinement of the VO model.

ing towards that goal. In other words, a member receives $1/N$ of the cost of the goal:

(∀ g,o. g ∈ goals ∧ o ∈ goalMembers0(g) ∧ card(goalMembers(g)) ≠ 0 ∧ finite(goalMembers(g)) ⇒
memberBudget(o) = GoalCost(g↦ goalMembers(g)) ÷ card(goalMembers0(g)))

As this division is carried over integers, we know that each member will receive equal share of the cost and that due to the remainder, the total cost is less than the sum of the individual member budgets. However, this error remains in practice small since the goal cost will be much larger than the number of participants. This can be forced even in cases of small goal costs by adjusting the measurement unit (e.g. the cost in Euros to the cost in Cents).

In the next event, Operation, an uncompleted goal, aGoal, is chosen as well as a set of member organisations such that this set is capable of achieving the goal (as defined by the goalMembers function). This goal is then added to the set of completed goals of the VO. Once the set of completed goals reaches the set of VO goals, the Dissolution event is triggered, which ends the VO lifecycle by setting the status goal to Stop.

At this level, we can define the following invariant, which expresses the $1/N$ cost-balancing policy using the delta distance measure.

Invariant 1 (All VO members have equal goal budgets) $\forall g, o1, o2.g \in goals \wedge o1 \in goalMembers(g) \wedge o2 \in goalMembers(g) \Rightarrow$
(memberBudget$(o1)$ $-$ *memberBudget*$(o2)$ $=$ *delta)* $\vee$
(memberBudget$(o1)$ $-$ *memberBudget*$(o2)$ $=$ *0-delta)* $\square$

This invariant states that the difference in member budget between any two organisations working on the same goal is only *delta* (or *–delta*) units away, where delta is a variable measuring the cost distance. The machine sets this variable to zero in order to implement the $1/N$ cost-balancing policy. However, other values are also possible, which would reflect incremental cost-sharing policies (similar to salary systems). As we mentioned above, the invariant is enforced thanks to the condition stating that each member will receive $1/N$ of the cost of a goal among N organisations working on that goal.

7 Goals, Organisations and Services

The second refinement, which represents our concrete model, is based on the concept of *services* and their relation to goals and organisations. The concrete machine and its context are shown in Figures 7. Based on this context, the second refinement, VORef2, of the VO machine is defined as in Figure 7.

The context, VBERef2, defines a new type called Services. These are the services advertised in a VBE. In addition to these, the context defines three relational valued constants. These are Requires, which models the set of services that a goal

MACHINE VORef2 **REFINES** VORef1 **SEES** VBERef2

VARIABLES
status, goals, completedGoals, goalCandidates, goalMembers, delta, memberBudget, memberServices

INVARIANTS
/∗**Define the type of memberServices**∗/
memberServices ∈ Organisations → ℙ1(Services) ∧

/∗**The 1/N cost-balancing policy invariant: sets of member services have equal costs**∗/
∀$g, o1, o2$. g ∈ goals ∧ o1 ∈ goalMembers(g) ∧ o2 ∈ goalMembers(g) ⇒
(ServiceCost($o1$)(memberServices($o1$)) − ServiceCost($o2$)(memberServices($o2$)) = delta) ∨
(ServiceCost($o1$)(memberServices($o1$)) − ServiceCost($o2$)(memberServices($o2$)) = 0-delta)

INITIALISATION
goals :∈ ℙ1(Goals) ∥ completedGoals := ∅ ∥ goalCandidates := ∅ ∥ goalMembers := ∅ ∥
delta := 0 ∥ memberBudget := ∅ ∥ memberServices := ∅ ∥ status := Id END

Identification REFINES Identification
/∗**Identify potential candidates**∗/
WHEN status = Id THEN goalCandidates := goals ◁ GoalCandidates ∥ status := Fr END

Formation REFINES Formation
/∗**Form the VO membership**∗/
ANY goalMembers0, memberBudget0, memberServices0 WHERE status = Fr ∧

/∗**Type of goalMembers0**∗/
goalMembers0 ∈ goals → ℙ1(Organisations) ∧ goalMembers0 ⊆ goalCandidates ∧

/∗**Type of goalBudget0**∗/
memberBudget0 ∈ Organisations → ℕ1 ∧

/∗**Type of memberServices0**∗/
memberService0 ∈ Organisations → ℙ1(Services) ∧

/∗**The definition of memberServices0**∗/
(∀g,o g ∈ goals ∧ o ∈ goalMembers0(g) ⇒ memberServices0(o) = Requires(g) ∩ Offers(o)) ∧

/∗**An extra condition on memberServices0:**
The cost of member services is ≤ their member budget∗/
∀ g,o. g ∈ goal ∧ o ∈ goalMembers(g) ⇒ ServiceCost(o)(memberService0(o)) ≤ memberBudget0(o) ∧

/∗**The 1/N cost-balancing policy condition**∗/
(∀g, o.g ∈ $goals$ ∧ o ∈ goalMembers0(g) ∧ card(goalMembers0(g)) ≠ 0 ∧ finite(goalMembers0(g)) ⇒
memberBudget0(o) = GoalCost(g ↦ goalMembers0(g))÷card(goalMembers0(g)))
THEN
goalMembers := goalMembers0 ∥ memberBudget := memberBudget0 ∥ status := Op END

Operation REFINES Operation
/∗**Pick an uncompleted goal and achieve it**∗/
ANY aGoal, memberSet WHERE status = Op ∧ completedGoals ≠ goals ∧
aGoal ∈ (goals\ completedGoals) ∧ memberSet = goalMembers(aGoal) THEN
completedGoals := completedGoals ∪ {aGoal} END

Dissolution REFINES Dissolution
/∗**When no more uncompleted goals, stop the VO**∗/
WHEN status = Op ∧ goals = completedGoals THEN status := Stop END

END

CONTEXT VBERef2 **REFINES** VBERef1

SETS
Services

CONSTANTS
Requires, Offers, ServiceCost

AXIOMS
Requires ∈ Goals → ℙ1(Services) ∧ Offers ∈ Goals → ℙ1(Services) ∧
ServiceCost ∈ Organisations → (ℙ1(Services) → ℕ1)

END

Fig. 7 The second refinement of the VO model.

requires, Offers, which models a set services offered by an organisation in a VBE and finally ServiceCost, which models the price of a set of services as advertised by an organisation in a VBE.

The concrete machine resembles the previous refinement except that an extra variable, memberServices, is introduced. This variable represents the service currently offered by the member organisations and used by the VO. The memberServices is given a value in the Formation event as a function from organisations to sets of services such that for any particular organisation, o, working towards a goal, g, then memberServices(o) is the set of services both required by g and offered by o. Our cost balancing policy imposes the same condition as in the previous refinement, which is that the budget received by each member organistion is equal to the total goal cost divided by the cardinality of the set of organisations working on that goal.

Now, we can state the following policy invariant at the level of services.

Invariant 2 (Sets of member services have equal costs) $\forall g, o1, o2.\ g \in goals \wedge o1 \in goalMembers(g) \wedge o2 \in goalMembers(g) \Rightarrow$
$(ServiceCost(o1)(memberServices(o1)) - ServiceCost(o2)(memberServices(o2)) = delta) \vee$
$(ServiceCost(o1)(memberServices(o1)) - ServiceCost(o2)(memberServices(o2)) = 0\text{-}delta)$ $\square$

The invariant states that the distance among sets of services belonging to one member is delta from the sets of services employed by another member towards the same goal. This invariant constitutes a more refined version of the invariant mentioned for the previous machine in the sense that equality among member budgets for achieving a goal is now propagated to the level of services resulting in the cost of all services offered by a member towards that goal being equal to the cost of all services offered by any other member working on the same goal.

8 Related Work

There is a fresh interest in the problem of policy refinement, given the complexity of dynamic distributed systems as envisaged in global computing. Bandara et al [8] uses a goal-oriented technique for policy refinement. In their work, a formal representation of a system, based on the Event Calculus [19], is used in conjunction with adductive reasoning techniques to derive the sequence of operations that will allow a given system to achieve a desired policy. An abstract policy is represented as a goal, and goal-oriented techniques are used to refine a policy into more concrete ones. Their approach differs from ours in that they assume the existence of a concrete architecture, which is expressed in UML and then translated to the Event Calculus.

In [22], Chadwick et al. propose a refinement approach for access control policies in Grids. Central to their approach is the existence of a hierarchy representing resources at different layer of abstractions. A policy is represented at the most abstract

layer, which is then refined into more concrete policies following the resource hierarchy. The hierarchy and the policies are specified using the ontological language OWL, so semantic-web reasoning is exploited to infer the concrete policies. Our work can be seen as a generalisation of their techniques in which the resource hierarchy and policies are generated simultaneously, exploiting the stepwise refinement approach.

Another line of work related to our is the formal modelling of Grids and VOs. Németh and Sunderam [18] define an operational model of grids and VOs based on the theory of ASMs [21]. They start first by defining a generic model that can be used to describe both distributed and Grid computing. This generic model consists of the universes of applications, processes, users, resources, nodes and tasks. These universes are related to one another through multiple mappings, which define the structure of systems. Our formal models can be seen as abstractions of their models.

The work of Janowski et al. [15] identifies two combinations of real-world enterprises that lead to the achievement of common goals. These are the extended enterprise and the virtual enterprise (which corresponds to the notion of a VO in our terminology). In the former, members of an extended enterprise satisfy one another's needs by matching the output of one member to the input of another. On the other hand, a virtual enterprise allows member organisations to cooperate and coordinate their resources and infrastructures in order to achieve the common goal. Hence, a virtual enterprise is a tighter coalition than an extended enterprise, which operates beyond the business interface of its members.

9 Conclusions

VOs are examples of distributed systems in which participants offer different kind of capabilities and resources in order to achieve common goals. Given the complex nature and rich state of this kind of systems, an incremental approach to build VOs is necessary. Here we have shown how to develop VOs and their policies using the refinement approach. We have also developed a similar model for refining other security-related policies [4], such as the Chinese Wall policy [9].

A key characteristic in our approach is to express system entities and their policies at the same level of abstraction. Then both components (i.e. entities and policies) are refined simultaneouly. The stepwise refinement approach allows one to build a system in an incremental way, adding at each step more concrete/operational detail. The refinement theory [6] guarantees the correctness of the whole approach and the existence of automatic tools [20] facilitates the verification process.

The use of the Rodin toolkit in discharging proofs and animating the models has been helpful in improving our understanding of the problem we are tackling (cost-balancing policy refinement) and its domain of application (VOs) in the sense that some assumptions made about the problem and/or its domain proved not to be valid.

As future work we plan to include in our model failure in achieving a goal. This would trigger the evolution sub-phase of the operational phase of the VO lifecycle, which is used to represent more dynamic behaviour in a VO.

References

1. Abrial, J.R.: The B Book. Cambridge University Press (1996)
2. Abrial, J.R. (ed.): Modeling in Event-B: System and Software Engineering. Cambridge University Press (2010)
3. Abrial, J.R., Mussat, L.: Introducing dynamic constraints in b. In: D. Bert (ed.) B98: Recent Advances in the Development and Use of the B Method, *Lecture Notes in Computer Science*, vol. 1393. Springer-Verlag (1998)
4. Arenas, A., Aziz, B., Bicarregui, J., Matthews, B.: Managing conflicts of interest in virtual organisations. Electron. Notes Theor. Comput. Sci. **197**(2), 45–56 (2008). DOI 10.1016/j.entcs.2007.12.016. URL http://dx.doi.org/10.1016/j.entcs.2007.12.016
5. Arenas, A.E., Djordjevic, I., Dimitrakos, T., Titkov, L., Claessens, J., Geuer-Pollmann, C., Lupu, E.C., Tiptuk, N., Wesner, S., Schubert, L.: Toward web services profiles for trust and security in virtual organisations. In: Collaborative Networks and their Breeding Environments (PRO-VE 2005). Springer (2005)
6. Back, R.J., , Wright, J.V.: Refinement Calculus: A Systematic Introduction. Springer-Verlag (1998)
7. Back, R.J., Sere, K.: Superposition refinement of reactive systems. Formal Aspects of Computing **8**(3), 324–346 (1996)
8. Bandara, A.K., Lupu, E.C., Moffett, J., Russo, A.: A goal-based approach to policy refinement. In: Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY 2004. IEEE (2004)
9. Brewer, D., Nash, M.: The chinese wall policy. In: IEEE Symposium on Research in Security and Privacy. IEEE (1989)
10. Camarihna-Matos, L.M., Afsarmanesh, H.: Elements of a ve infrastructure. Journal of Computers in Industry **51**(2), 139–163 (2003)
11. Camarihna-Matos, L.M., Afsarmanesh, H., Ollus, M.: Ecolead: A holistic approach to creation and management of dynamic virtual organizations. In: Collaborative Networks and their Breeding Environments (PRO-VE 2005). Springer (2005)
12. Consortium, R.: Event B Language. Technical Report, Deliverable D7 (2005). Available at http://rodin.cs.ncl.ac.uk/deliverables/rodinD10.pdf.
13. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. International Journal of Supercomputer Applications **15**(3) (2001)
14. Golby, D., Wilson, M., Schubert, L., Geuer-Pollmann, C.: An assured environment for collaborative engineering using web services. In: Proceedings of the 2006 conference on Leading the Web in Concurrent Engineering: Next Generation Concurrent Engineering, pp. 111–119. IOS Press, Amsterdam, The Netherlands, The Netherlands (2006). URL http://dl.acm.org/citation.cfm?id=1566652.1566674
15. Janowski, T., Lugo, G.G., Hongjun, Z.: Composing enterprise models: The extended and the virtual enterprise. In: Proceedings of the Third IEEE/IFIP International Conference on Intelligent Systems for Manufacturing: Multi-Agent Systems and Virtual Organizations. IEEE (1998)
16. Moffett, J.D., Sloman, M.S.: Policy hierarchies for distributed system management. IEEE Journal of Selected Areas in Communications, Special Issue on Network Management **11**(9) (1993)
17. Morin, C.: Xtreemos: A grid operating sytem making your computer ready for participating in virtual organizations. In: 10th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC 2007). IEEE (2007)

18. Németh, Z., Sunderam, V.S.: Characterizing grids: Attributes, definitions, and formalisms. Characterizing Grids: Attributes, Definitions, and Formalisms **1**(1), 9–23 (2003)
19. R.A.Kowalski, M.J.Sergot: A logic-based calculus of events. New Generation Computing **4**, 67–95 (1986)
20. RODIN Consortium: Specification of basic tools and platforms. Technical Report, Deliverable D10 (2005). Available at http://rodin.cs.ncl.ac.uk/deliverables/rodinD10.pdf.
21. S. Dexter, P.D., Gurevich., Y.: Abstract state machines and schoenhage storage modification machines. Journal of Universal Computer Science **3**(4), 279–303 (1997)
22. Su, L., Chadwick, D.W., Basden, A., Cunningham, J.A.: Automated decomposition of access control policies. In: Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY 2005. IEEE (2005)
23. Wasson, G., Marty, H.: Toward explicit policy management for virtual organisations. 4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY2003) (2003)