Modelling Virtual Organisations: Structure and Reconfigurations

Stephan Reiff-Marganiec and Noor J Rajper

Department of Computer Science, University of Leicester, Leicester, UK
{srm13, nr76}@le.ac.uk

Abstract. Organisations have to adapt rapidly to survive in today's diverse and rapidly changing environments. The idea of virtual organisations emerged as an answer. There is a strong need to understand virtual organisations (VOs) in a formal way: changes can have side effects and hence one might wish to understand precisely what consequences a change might have. The Virtual Organisation Modelling Language (VOML) consists of sub-languages to model different aspects of VOs such as their structure or operational models: VO-S deals with structural aspects while VO-R addresses reconfigurations. The concepts are exemplified through a travel booking VO that needs to cope with extra demands imposed by a large event such as the Olympic games.

Keywords: Virtual Organisations; Structural Model; Reconfigurations

1 Introduction

Changing business environments and a diversity of emerging requirements make entities' (organizations, individuals, etc.) survival and success dependant on their ability to adapt dynamically to changing operating conditions [4, 6, 8]. VOs provide a solution when it is impossible for any single organization (particularly small or medium size enterprises; SMEs) to keep up with the speed of change in the environment and the diversity of demands prevailing in the market. Adaptability and diversity are some of the main defining characteristics of VOs – adaptability allows to cope with changing demands while diversity is multidimensional (e.g. the range of involved stakeholders, the offerings made, cultural and geographic positioning, etc.).

A VO is a loosely bound consortium of organisations that together address a specific demand that none of them can (at the given time) address alone – once the demand has been satisfied the VO might disband. A VO is created in the context of a VO Breeding Environment (VBE), which is a more stable grouping of business partners, but without a specific current goal [4]. Some VOs persist over longer time, but adapt to realign themselves to address changes in demand.

A VO has a structure that defines the tasks it is conducting to achieve its goals as well as defining which partners are involved. There is a need to rigorously define this structure, as it is the foundation to describing and analysing the reconfigurations of the VO [3], which are the way in which a VO adapts to changing demands.

In this paper we present VOML (the VO Modelling Language) focusing on two aspects: the modelling of the structure of a VBE/VO and the description of reconfigurations. For the former we present a new language that captures the key aspects of the VO at a level of abstraction meaningful to business users by making key concepts first class citizens of the language but still concrete enough to be mapped into more rigorous languages for operational models and rigorous reasoning and analysis. For the latter we introduce VO-R, the VO reconfiguration language. VOs can adapt in two fundamental ways: The ultimate goal of the VO might change, i.e. the VO should address a demand unrelated to its original purpose, in which case a new VO should be formed, possibly including current members. However, often the VO needs to adapt in more subtle ways to deal with fluctuations in capacity demands or alternative ways of achieving a task – it is this aspect which is addressed by VO-R.

The paper is organized as follows: Section 2 presents an overview of the VOML approach for modelling VOs, sections 3 and 4 introduce the structural and reconfiguration languages. The paper is rounded off with a discussion, related work and some concluding remarks and future directions in the final sections.

2 Overview of Approach

The Virtual Organization Modelling Language (VOML) is dedicated to VO development. VOML operates in the context of a VO Breeding Environment; VOs are formed dynamically to provide high-level functionalities by sharing a number of resources in a distributed way. The VOML approach [2] supports the definition of a structural and behavioural model of a fixed VBE based on three different levels of representation: (1) the definition of the persistent functionalities of the VBE; (2) the definition of the transient functionalities of the VOs that are offered by the VBE at a specific moment in time (a business configuration of the VBE) and (3) the ensemble of components (instances) and connectors that, at that time, deliver the services offered by the VOs present in the business configuration (a state configuration).

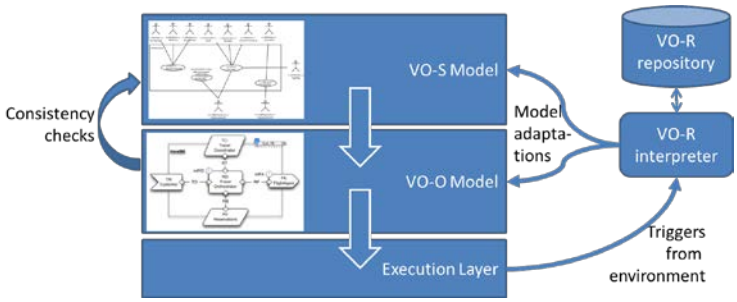

Fig. 1: Overview of Framework

Very briefly, structural models present the general structure of the VO (or VBE); these are mapped semi-automatically (semi as some refinement is required) into operational models. The operational models are quite close to respective execution frameworks (such as agent based systems or service oriented systems) and can be

mapped to these. The execution framework is monitored and any changes are reported back to the model levels where policies are activated to refine the VOs and ensure that they remain competitive. Reconfiguration rules are checked for consistency and furthermore any VO-model should be correct by construction (that is it should be a true refinement of the respective structural model) – however this is further ensured by consistency checks. Figure 1 depicts this overview in a graphical manner.

3 Structural Models: VO-S

The VO Structural Modelling Language (VO-S) defines the basic structure of the VO, its constituent elements, abstract process, and other details which define the essential structure of the VO and provide the basis for its operational models. The VO structural model consists of five basic elements: (1) Members, (2) Process, (3) Tasks, (4) VBEResource and (5) Data-Flow. Of these, Members, Tasks and VBEResources are elements that can occur in VBE specifications, too.

Members can be of one of three types: *Partners* are permanent members of the VBE, *Associates* are transient members of the VBE who have joined temporarily based on demands of some VO which requires some capability for which there is currently no member available in the VBE and finally *extEntity* which are transient members of a VO and are discovered for each VO instance and leave the VO when the instance finishes its life. For the travel booking VO, the hotel provider is a partner, while FlightBooking is an extEntity.

Process describes the workflow which leads to meeting the requirements of the customer of a VO at the highest level of abstraction. It lists only those tasks that contribute directly towards achieving the goals of the VO.

Task specifications define the competencies required by the VO from its members. They help in deciding the kind of restructuring that a task and hence a VO can undergo, having effects ranging from VO topology to member relationships. Tasks are complex and we will here only show some key features. Attributes exist in the *TaskScope* (where they provide a domain of discourse for configurations) and in the *ConfScope* (where they describe the current configuration). E.g. the `performedBy` attribute in the *TaskScope* might specify that the task can be performed by Associates and Members, the *ConfScope* would make precise who is performing the task in the current configuration. Also, each tasks specifies the competencies required for performing it: which *capability* does a performing partner need and what *capacity* is needed (e.g. a partner needs to provide accommodation, and 50 rooms are needed).

Reasons for forming virtual organizations are (a) an entity (organization) lacks some capabilities to take on a job; and (b) an entity has all the capabilities required but lacks the required capacity (often the case with Small and Medium organizations (SMEs) competing for large jobs). VO-S addresses this by offering three types of tasks: *AtomicTask* (tasks to be performed by only one member), *ReplicableTask* (tasks that can be shared to gain extra capacity) and *ComposableTask* (tasks that can be shared to address capability issues). Members can compete or cooperate *ReplicableTask*s.

VBEResource are resources that are owned by the VBE and made available to all VOs that stem out of the respective VBE. The VBE will have rules on their usage.

Data-Flow specifies the relationship between data items to assist with realizing concrete orchestrations, transitions and wires in operational model. It expresses which data items are expected from the customer and partners and their flow between tasks.

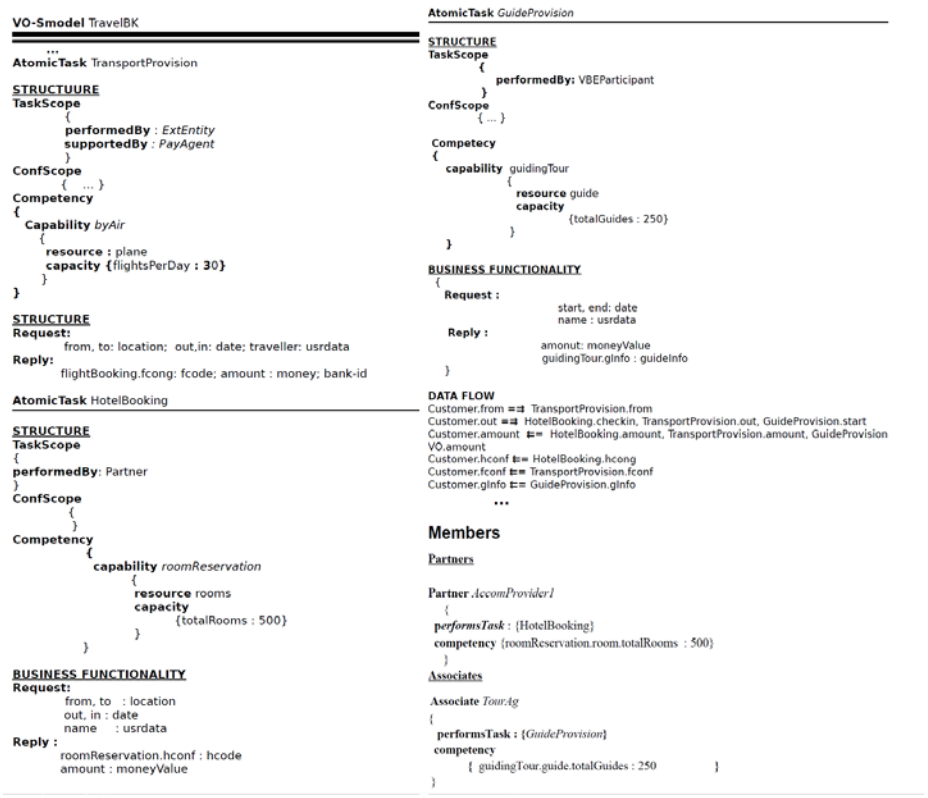

Fig. 2: VO-S Structural Model of the TravelBK VO

The TravelBK VO (Fig. 2) specifies a VO that is concerned with offering trip itineraries that include transport via air, accommodation and local excursions. There are two members in the VO (the AccomProvider1 is a VBE member, the TourAg is an associate). Air transport is provided by suitable external entities. All three tasks are atomic meaning each is performed by a single member.

4 Reconfigurations: VO-R

Policy languages have been successfully used for dynamically adapting systems. In this work we are particularly interested in policies which allow to reconfigure (adapt) a VO to cope with the changes that it is subjected to. We require and wish to model replication of tasks so that a number of members can combine capacity (either jointly or in competition) and decomposing complex tasks into smaller ones, so that several

members can combine their skills. Additionally we like to be able to change the membership of the VO by adding or removing members.

While policies provide a feasible approach towards reconfiguration, a special policy language is required which provides explicit constructs suitable for the domain of VO reconfigurations. Typically a policy language can be completely domain specific or a more generic ECA (event-condition-action) language can be adapted to new domains by providing appropriate actions, triggers and conditions. We adapt the APPEL [1] language – APPEL had been developed with a clear separation of domain and core language. APPEL has a style closer to natural language as it was aimed at non-technical end users, rather than developers or system administrators, something clearly of benefit the users of VO modelling languages. APPEL was originally aimed at call control, but has since been specialised for e.g. sensor networks or elderly care.

Policy rules are the basic building block of APPEL. Each rule consists of an optional trigger, optional condition and an action. A rule is applicable when its trigger has occurred (if one is defined) and its conditions (on the state and possibly trigger parameters) are satisfied (with the empty condition being trivially satisfied). Actions have an effect on the system to which the policies are applied. Policy rules can be grouped into larger policies. A typical APPEL policy looks as follows:

```
policy policy-name appliesTo task-id/member-id/VO-id
when [trigger(s)]
if [condition(s)]
do action(s)
```

For VO-R we have to add a number of triggers, conditions and actions. We also need to consider the meaning of the localisation: **`appliesTo`** allows to 'locate' a policy. In distributed settings this would allow to specify the location of a policy, for VOs this allows to express which task(s), member(s) or VO(s) a policy applies to. The specialization of APPEL to Virtual organizations forms the Virtual Organisations Reconfiguration language VO-R. There are several triggers (Table 1) and actions (Table 2) specific to the domain, some of which are shown below – we do at this stage not claim that this list is complete and ongoing investigation studies this aspect.

Table 1. Sample triggers for VO Reconfigurations.

Trigger	Description
`NoMemberWithRequiredCapacityFound`	Arises when no existing VBE member is found to provide suitable capacity (either because they are not willing to provide more resources to the VO or because they cannot).
`CapacityDeficit`	The VO has all required capabilities, but is lacking capacity to satisfy current demand by a task. The task could be conducted, but with limitations.
`CapabilityDeficit`	The VO lacks a required capability. A capability deficit means that certain tasks cannot be conducted.

Table 2. Sample actions for VO Reconfigurations.

Action	Description
`MakeTaskReplicable`	Changes the type of a task so that it can be shared between

`(task-id, allowedMembers, relationship)`	members. The resulting type will be a ReplicableTask that can be shared between at most `allowedMembers` members who compete or cooperate (depending on `relationship`).
`AssignTask(member-id, task-id)`	Assigns a member to a task. If the task is atomic or was unassigned this member will be the one conducting the task; if the task is replicable or composable this member will perform some part of the task.
`AddNewMember(member-id)`	Adds a new members to the VO.

Let us now consider two examples based on the TravelBK VO, which was presented earlier in detail. Specifically the VBE management has decided that TravelBK has a big role to play during a big upcoming event, such as the Olympic games, but needs to adapt to cope with the demands.

Scenario 1: *More hotel beds are needed than the current provider can provide.* The Olympic games will bring an influx of people who require accommodation. The current member responsible for the HotelProvision task has limited capacity and no other member can individually meet the expected demand, but each provider could contribute to the demand. The following reconfiguration policy makes the HotelProvision task shareable between up to 3 members; the allocation will be given to the cheapest one.

```
policy MoreBeds appliesTo HotelProvision
when NoMemberWithRequiredCapacityFound
do MakeTaskReplicable(HotelProvision,3,[competition,cheapest])
```

Scenario 2: *One of the hotel partners has had a fire and had to withdraw their commitment.* Sadly one of the hotels had a fire and cannot provide the promised accommodation. This is a kind of contract violation in the eyes of the VBE – clearly in business cases this happens and the VO is prepared to react to it as it has the following rule (this rule assumes that memberX is only involved in TaskY). Before a member can be removed the tasks assigned to that member need to be removed. The application of the rule might lead to a follow-on problem of a capacity or capability deficit (a capacity deficit was shown in Scenario 1 and capability deficits are similar).

```
policy MemberQuits appliesTo memberX
do UnAssignTask(memberX, TaskY)
  andthen RemoveMember(memberX)
```

5 Related Work

VO's have been modelled using a number of generic modelling frameworks, such as UML, ERP or PetriNets. For example, in [3] a VO is modelled using the Vienna Development Method in five dimensions: membership, information representation, provenance, time and trust. The formal model allows for analysis and verification. Our modelling language raises the level of abstraction to a point where it is possible to directly support notions and concepts that are paramount in the domain of VO such

as members and resources. The use of UML stereotypes to achieve the same could be investigated, and indeed would be interesting to enhance our work with a graphical syntax – VO-R could then be expressed through graph transformation rules.

Like our proposal, [12] allows tasks to be shared, however they only allow sharing of a task based on capacity whereas we allow sharing based on capabilities as well. Additionally we cover the effects of such sharing on the members and on the process model. Besides, [12] is based on Agent Technology whereas our work is at a higher level of abstraction than a platform or technology specific solution.

The field of dynamic adaptability in general [7] is relevant. ASSL [13] in particular focuses on flexibility by providing a modelling language for autonomous systems. ASSL abstracts away from business level requirements of the system, however these are at the heart of VO modelling. More in the domain of VOs, the Arcon framework provides a reference model at an abstract level, while we attempt to make precise more of the structural and operational aspects of the VO and its reconfigurations.

In many areas of computing policy languages have become widely accepted as a way of reconfiguring systems to cope with changes in the environment [13]. Often policy languages describe security or low level system management aspects [10]; at higher levels of abstraction the APPEL [1] policy language has been adapted for workflows, where it allows to insert and delete tasks [9]. For VO's no such language currently exists but the presented reconfiguration language is a variant of APPEL.

6 Conclusions and Future Work

VOs capture organizations that are formed as relatively short lived consortia from other organizations. The ability to dynamically adapt to diversity in demands extends the lifespan and profit of VOs. In this paper we put forward a novel language to allow for modelling the structure and reconfigurations of VOs. VO-S allows to describe the structure of a VBE or a VO in terms of the key elements such as tasks or partners, while VO-R allows to specify rules that describe how a VO should adapt if asked to do by the VBE or if events occur from the VO's environment that need to be addressed. The VBE might stipulate changes when it sees them as beneficial to the business; the environment might raise triggers e.g. a capability or capacity shortage.

As reconfigurations are a core aspect of VOs, other approaches provide attempts at dealing with similar issues. Our approach is quite close to how real organisations behave in that we are providing a flexibility by allowing reconfiguration of tasks into a different task type by *replication* or *decomposition*. It is crucial to understand the changes as they can range from the VO architecture (its structure) to relationship between members and further more to changes in control and data flow. Existing work often focuses on replication of resources rather than on the members, however it is usual that new capacity or capability comes from new partners.

Our modelling and reconfiguration is close to the business requirements as shown in the reconfiguration scenarios and the respective policies – the policy language is intuitive and staff with business knowledge can write the required policies (actually the previous uses of APPEL in telecommunications were targeted at lay end-users).

Finally, we have clearly separated changes that change the tasks and hence can have an affect on the goal of the VO from those that make it more competitive in its

environment. Changes to goals are achieved by new VOs being created from the VBE, while the presented reconfigurations allow for adaptations that maintain the overall goal. This distinction is critical, as it allows specification of properties that the VO needs to achieve or preserve and ensure that these are indeed adhered to.

In future work we will analyse whether VO-R can be lifted to the VBE level, to see if 'changes' to the structure that occur during the creation of new VOs can described in the same way (e.g. rules on selection of VO partners). Initial analysis shows that this is feasible, but will require further domain specific actions and triggers in VO-R. We will also consider performance modelling based on variants of VO-S models to understand the affect of VO-R rules on adaptability of the VO and more formal detection of conflicting rules, based on the formal semantics of APPEL [11]. These types of analysis will lead to an understanding that is useful for any VO.

References

1. Blair, L., Reiff-Marganiec, S., Turner, K.: APPEL: The ACCENT Policy Environment/Language. Technical report CSM-164. Department of Mathematics and Computer Science. University of Stirling. (2005)
2. Bocchi, L., Fiadeiro, J.L., Rajper, N., Reiff-Marganiec, S.: Structure and behaviour of virtual organisation breeding environments. In: Bryans, J., Fitzgerald, J.S. (Eds.) FAVO. EPTCS, vol. 16, pp. 26–40 (2009)
3. Bryans, J., Fitzgerald, J., Jones, C., Mozolevsky, I.: Formal modelling of dynamic coalitions, with an application in chemical engineering. In: Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. pp. 91–98. (2006)
4. Camarinha-Matos, L.M., Silveri, I., Afsarmanesh, H., Oliveira, A.I.: Towards a framework for creation of dynamic virtual organisations. In: in L.M. Camarinha-Matos et al. (Eds.) 6th IFIP Working Conference on Virtual Enterprises PRO-VE 2005. pp. 26–28. Springer (2005)
5. Camarinha-Matos, L.M. and Afsarmanesh, H.: Collaborative Networks: Reference Modeling. Springer US. 2008.
6. Cummings, J., Finholt, T., Foster, I., Kesselman, C.: Beyond being there: A blueprint for advancing the design, development, and evaluation of virtual organizations. Final Report from Workshops on Building Effective Virtual Organizaions (2008)
7. Di Marzo Serugendo, G., Fitzgerald, J., Romanovsky, A., Guelfi, N.: A metadatabased architectural model for dynamically resilient systems. In: SAC07 (ACM Symposium on Applied Computing). pp. 566–572. ACM (2007)
8. Foster, I.: The anatomy of the grid: Enabling scalable virtual organizations. International Journal of Supercomputer Applications 15 (2001)
9. Gorton, S., Montangero, C., Reiff-Marganiec, S., Semini, L.: StPowla: SOA, policies and workflows. In: Nitto, E.D., Ripeanu, M. (eds.) ICSOC Workshops. Lecture Notes in Computer Science, vol. 4907, pp. 351–362. Springer (2007)
10. Lupu, E., Sloman, M.: Conflicts in policy-based distributed systems management. IEEE Transactions on Software Engineering 25(6), 852–869 (Nov/Dec 1999)
11. Montangero, C., Reiff-Marganiec, S., Semini, L.: Logic-based conflict detection for distributed policies. Fundamenta Informaticae 89(4), 511–538 (2008)
12. Norman, T.J., Preece, A., Jennings, N.R., Luck, M., Dang, V.D., Nguyen, T.D., Deora, V., Shao, J., Gray, W.A., Fiddian, N.J.: Agent-based formation of virtual organisations. Knowledge Based Systems 17, 2004 (2004)
13. Vassev, E., Hinchey, M.: ASSL: A software engineering approach to autonomic computing. IEEE Computer 42(6), 90–93 (2009)