Animation-based Validation of a Formal Model of Dynamic Virtual Organisations

J. S. Fitzgerald, J. W. Bryans, D. Greathead, C. B. Jones and R. Payne
School of Computing Science, Newcastle University, UK
John.Fitzgerald@ncl.ac.uk.uk

Abstract

We describe a study into the use of animation in the analysis of a formal model of information flow in dynamic virtual organisations (VOs). A generic formal model of a VO structure composed of autonomous agents sharing information, developed in VDM, was animated via the VDMTools interpreter running an application-specific script. A user interface allows domain experts to interact with the model without being directly exposed to the formalism. In a case study, a domain expert interacted with the model under observation and debriefed. The form of the study allows us to draw conclusions about the suitability of such formal models for exploring the design of policies governing VOs.

Keywords: Formal Methods, Virtual Organisations

1. INTRODUCTION

Advances in network technology, software and services are making it ever easier for organisations and individuals to form loosely-coupled temporary alliances in order to take advantage of a business opportunity or respond to an acute crisis. Such alliances are often called Virtual Organisations (VOs), virtual enterprises or dynamic coalitions. VOs are often dynamic: they evolve as the environment changes, as resources become available or unavailable and members join and leave. Although each VO is unique, they share some common features such as dynamically changing membership, mechanisms for information transfer and authorisation structures. In many applications, the ability to analyse system-level properties such as information flow, security and privacy is particularly significant. However, the architects of such coalitions currently lack a basis on which to evaluate at design-time the consequences of the decisions that they make regarding coalition architecture and policies.

Formal methods offer one technology for analysing the properties of potentially complex systems like dynamic VOs. Faced with the wide range of systems that qualify as VOs, we used formal modelling techniques to map out a space of VO architectures on the basis of "dimensions" including membership, information representation, provenance, time and trust [2]. Each dimension represents a choice, e.g. of a particular membership policy, information representation etc. Thus a particular VO might be described in terms of the choices made in each of these dimensions. We developed formal models representing specific architectural decisions within each dimension and validated the approach by showing how it could characterise a VO architecture developed for collaborative development in the chemical engineering industry [3]. The process of characterising and formalising aspect of this particular VO raised several issues, notably in handling VO initiation and dissolution.

Although formal modelling might give a useful basis for discussing VO structures, our industrial experience with formal modelling and analysis technology suggested that much of the value of formal modelling comes when domain experts interact with models directly. In 2007, we had an opportunity to perform a study in collaboration with the UK Defence Science and Technology Laboratory (Dstl), who have an interest in tools that will assist policy evaluation and design in dynamic coalition environments. Through this collaboration with an end user, we sought to answer the following questions, among others:

Can we provide formal models and tools that help in architecting VOs and their related policies?
Our previous work suggested that an executable model could be configured with particular decisions

relating to the dimensions of interest. Scenarios could be played out in this instantiated model in order to analyse the consequences of (combinations of) policy decisions.

Are users' perceptions of information flow in VOs consistent with such a model? Human decision makers in VOs should have an accurate picture of the state of the real VO and hence the consequences of their decisions. User errors are one of the most common reasons for (socio-technical) system failure. This is recognised in the context of military information systems in [9] where it is argued "The rewards achieved through good training are best achieved at the Human-Machine Interface level". Furthermore, the advent of the UK Network Enabled Capability has raised many issues with regard to information management [8].

In the remainder of this paper, we describe a short study, conducted with industry users, to give initial answers to these questions. Our approach has been constructive; to develop an exploratory environment that allows domain experts who are not familiar with formal modelling notations the opportunity to experiment with alternative policy decisions on models of VOs. We first describe the modelling approach used (Section 2), the exploratory environment (Section 3) and then outline the case study undertaken with users in which a specific VO model is evaluated against a scenario (Section 4). We present conclusions and suggest further work (Section 5).

2. MODELS OF VIRTUAL ORGANISATIONS

The VO models used in our work were developed using the Vienna Development Method (VDM) technologies. VDM is a model-oriented formalism with a history of strong tool support and a record of application in industrial settings that are new to the adoption of formal techniques. A theme in VDM-based research has been that of lowering the "entry barrier" to the industrial use of formal methods without compromising the level of rigour in the underlying formalism [6]. A major part of this has been the provision of robust tools (VDMTools). The VDM-SL Specification Language [5] has been extended to handle object-oriented structuring and concurrency (VDM++ [7]), distribution and real-time (VICE extensions [11]). In this work, we have used both VDM-SL and VDM++. For details of the formalism, readers are referred to the texts and the VDM Portal: `http://www.vdmportal.org`. The VDM modelling languages have a very large executable subset, and VDMTools contains an efficient interpreter, allowing models to be evaluated directly by testing them on user-defined scenarios.

VDM models are usually based on descriptions of system state given in terms of variables each of which belongs to a type constructed from abstract base types and type constructors such as records, sets, sequences and finite mappings. Types may be restricted by data type invariants. For example, a very simple VO model might have two state variables describing respectively a collection of coalitions and agents, defined as follows. Coalitions are modelled as finite sets of agents.

```
coals : map Cid to set of Aid
agents : map Aid to Agent
inv mk_VOSpace(coals, agents) ==
  dunion rng coals subset dom agents and
  {} not in set rng coals
```

The invariant in this model is a predicate requiring that only known agents participate in coalitions and that no coalition is empty. The final conjunct embodies the constraint that no coalition (or virtual organisation) has a life independent of its members. Thus this model does not admit a "secretariat" level within VOs.

Functionality is described in VDM by operations specifying relations over the state variables, inputs and results. Operations may have side effects on state variables and may be defined with the aid of auxiliary functions. At the most abstract level, operations are specified in an implicit style by pre- and postconditions. For example, the following operation specification describes the removal of an agent from a coalition. The " " suffix on a state variable indicates the value in the variable before execution of the operation; postconditions are thus predicates of two states.

```
Remove (a:Aid,c:Cid)
ext wr coals : map Cid to set of Aid
pre c in set dom coals and a in set coals(c)
```

```
post coals = coals~ ++ {c |-> coals~(c)-{a}}
```

In VDM, proof obligations embody internal consistency constraints on models. For example, the *satisfiability* proof obligation requires that any invariants on state variables are preserved by operations. In the example above this condition would be broken: there is no constraint to ensure that the coalition `c` is not being emptied of its last member. This rigorous approach is one means by which questions are raised during the definition of a VO infrastructure, as we found with the chemical engineering example. In this case, on consulting with the engineers involved, it became clear that special procedures should be put in place to dissolve VOs, for example in order to ensure that common VO information assets are not lost.

The model introduced above deals only with the membership aspect of a VO. Other models [2] have been developed for other aspects. For our study with Dstl, we used two VO models which both allowed information to be stored at the level of the VO as well as within the participant agents. The two models different in the operation that added members to the VO: in one model, termed the *full disclosure* model, all the VO-level information is downloaded into agents when they join; in the other *partial disclosure* model, new members do not automatically find out the information already placed in the VO-level repository.

3. AN EXPLORATORY ENVIRONMENT

The purpose of our exploratory environment was to provide a facility for domain experts to interact with a VO model without requiring familiarity with the notation. This is done primarily by running a scenario which invokes the operations in the VO model, running them on the VDMTools interpreter.

3.1. Environment Architecture

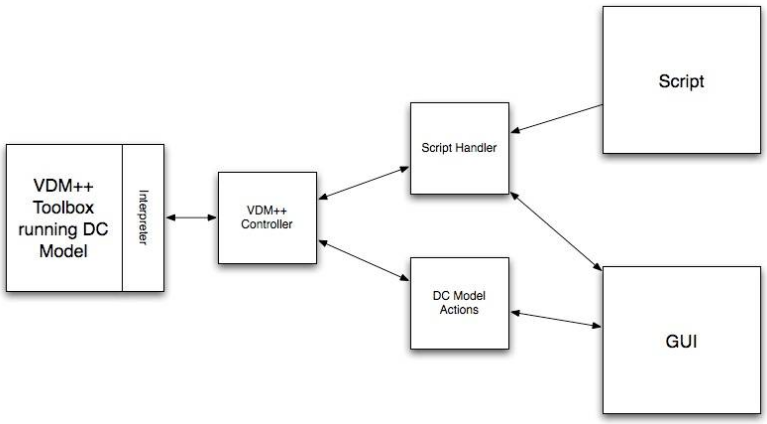

Figure 1: Outline architecture of the VO exploratory environment

Figure 1 is a simplified view of the environment's architecture, based on a Model-View-Controller [4] architecture. The VDM++ model contains information regarding agents, the coalition structures and the information in the scenario. The view is provided by a Java graphical user interface (GUI) which allows the user to view the data in the model. The scenario script and Java controller classes prompt and manage interaction from the user via the view, and pass any changes of state data to the model.

In order to support analysis by execution in the interpreter, directly executable versions of the operations are constructed in VDM. These retain the pre-/postconditions (which can be used as run-time checks). For example, the `Remove` operation above might be defined explicitly as follows, with a body containing an assignment to the state variable.

```
public Remove : Aid * Cid ==> ()
Remove(a,c) == (coals := coals ++ {c |-> coals(c) \ {a}})
pre c in set dom coals and
    a in set coals(c)
post coals = coals~ ++ {c |-> coals~(c) \ {a}};
```

The scenario is encoded as a script in Java. This makes calls to the VDMTools application programmer interface through which commands are passed to the interpreter running the VO model. For example, the following Java source invokes operations to create a new agent in the VO model and add it to a VO called the "response coalition":

```
evaluateExpression("t.scenario.CreateNewAgent(t.Unit,?) ");
Gui.updateAgents;
evaluateExpression("t.scenario.Join(t.Unit,t.response_coalition")
```

The execution of the Java causes the interpreter to execute the following explicitly-defined operations in the VO model:

```
CreateNewAgent: Aid ==> ()
CreateNewAgent(a) == agents := agents ++ {a |-> mk_Agent({},{})} ...

Join: Aid * Cid ==> ()
Join (a,c) ==  coals := coals ++ {c |-> ...} ...
```

3.2. User Interface and Scenario

The GUI relays state data from the model to the user. It was developed using Java Swing and AWT libraries, giving us a platform-independent interface with the native look and feel of the operating system. The GUI consists of a tabbed pane containing the views onto the model (Fig. 2)

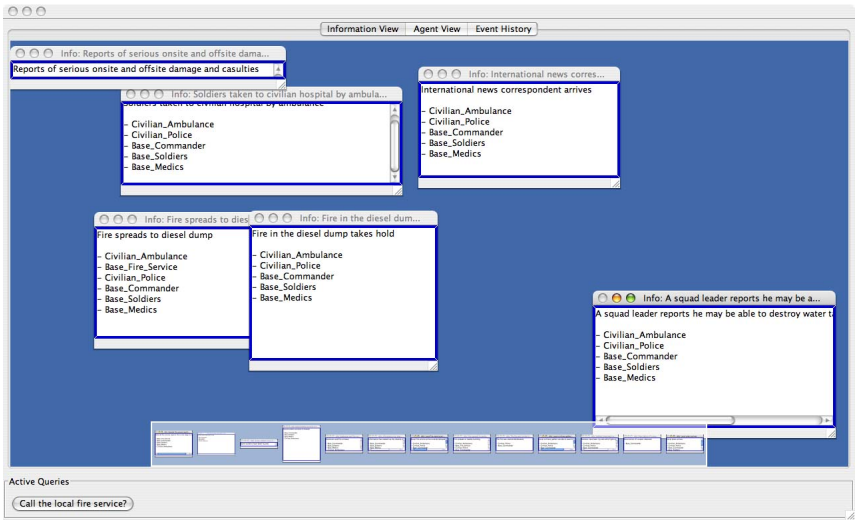

Figure 2: User interface to VO exploratory environment

The interface can show the model from two perspectives. The *omniscient perspective* reveals the full state of the model in terms of agents and the information that they contain. The *participant perspective* shows the point of view of a participating agent, showing only the information that has been the agent has derived for themselves or received from others. Each perspective may be examined from two main views. The *information-oriented* view provides one window for each piece of information presented to the user, indicating which agents are currently known to hold the information. The *agent-oriented* view shows each agent and the information items they hold. Coalition membership is represented using colour coding.

During operation the interface allows the user to manage information presented on the screen in the usual ways by moving, resizing, minimising and maximising windows. The arrangement is preserved when switching between views. In our study, the user took the participant perspective during execution of the scenario. After completion of the scenario, they were invited to use the omniscient perspective to see whether the distribution of information in the VO corresponded to their expectations.

The exploration environment that we developed was evaluated by means of a study in which the user is invited to make an initial policy choice between partial and full disclosure. The model corresponding to the

relevant policy decision is initialised and animated on the interpreter using a scenario that was developed in conjunction with domain experts.

The interface is driven by the controller script which interface to display new items of information and questions (decision points) for the user. The user simply acknowledges new information but may respond to decision points from a range of alternatives offered by the script. Each piece of information, and all decisions made by the user are recorded in an event history log which is saved for reference after completion of the scenario.

The scenario was developed in consultation with a domain expert from Dstl. It involves the management of an acute crisis (a fire) and the news that the fire was caused maliciously. The user plays the role of the manager of the site in which the fire occurs. Decision points built into the scenario concern the opening of the site to emergency services, the management of staff and the press. The study is primarily concerned with managing the membership of a single emergency response VO and the flow of information between participants in that VO. The user makes their own decisions about the value of information presented to them.

The scenario script includes rudimentary calculations to model processes in the environment such as spread to the fire or numbers of casualties. It also contains the possibility of an unanticipated "leak" of information from within the VO to agents outside. After completion of the scenario from the omniscient perspective, the user was invited to explore the state from the participant perspective, note the information leak and perhaps re-run the scenario to see if any of the decisions made might avoid the leak.

4. CASE STUDY

A case study [10] uses an individual or a small number of participants who are closely observed. It tends to be exploratory in nature. The main purpose of our case study was to assess the model exploration environment, observe and record user behaviour, and seek to identify aspects worthy of further study.

In informal initial studies, several participants used the environment to work through the scenario, allowing us to evaluate and debug the environment before a formal final case study. In this final study, the participant was an expert with military command experience experience. He ran the scenario several times, his behaviour was observed throughout and he took part in informal exploratory interviews after each run. The interviews were conducted in a non-directive way, allowing the participant to give their own thoughts rather then being led to answer specific questions.

Several useful observations, notably about the desire for an appropriate non-textual presentation of the context information (e.g. a map showing locations of fire, fire units etc.) The user was keen to have a less constrained palette of actions available to him during the execution of the scenario. We had fought shy of this in developing the environment because we did not want to reveal too much about the structure of the formal model to the user. Future work will aim to address this.

Following the study, the user was very positive about the value of having a model that could be explored in this interactive way. Furthermore, the formality of the model was seen as a considerable benefit in opening the possibility of specialist, rigorous analysis.

5. CONCLUSIONS

We initially asked whether we can provide formal models and tools that help in architecting VOs and their related policies. We believe that the case study developed here gives some support to this proposition. In the case study, we have shown how it is possible to explore the consequences of alternative policy decisions in a way that is accessible to domain experts. We also asked whether users' perceptions of information flow in VOs would be consistent with a model. This is a rather general question, but the case study has demonstrated that it is possible to explore the interaction between participants in a VO in order to assess the validity of models developed. One promising area of exploration is the coupling of access control policy models with those of VO membership [1], allowing VO participants to adjust the access rights granted to VO members as the VO and its goals evolve.

The scenario itself contains a model in the form of the variables that represent the environment state. This could be separated out into a model that could be coupled to that of the VO. In many cases, environment processes such as the spread of a fire are already described by well-tried specialist models (rarely formal). Furthermore, it is certainly possible to widen the scope of the exploratory environment to allow multiple users to take the roles of different concurrent actors in the same scenario.

We have hardly begun to exploit the benefits that accrue from the use of a formal model. In particular we have only animated the model and have not used any formal static analysis beyond generating proof obligations. As we indicated at the outset, our aim is to have a low barrier to entry into the formalism. Specifically, we wish to show that it is possible to gain benefit from an early commitment to the use of formal modelling. When formal models are developed after a proof-of-concept implementation, that basic implementation often becomes the basis for the product and the model is left lagging behind, before eventually being abandoned.

In future work, we aim to provide proof-based validation of system-level VO properties, noting that the key challenge here is eliciting and formalising these properties. In a domain as weakly defined as that of network-enabled virtual organisations, it is to be hoped that exploratory environments such as these will allow the use of formal models from very early stages when the key abstractions underpinning the expression of system-level properties are as yet uncertain.

Acknowledgements: We are grateful to Tom McCutcheon, Helen Phillips and Olwen Worthington of Dstl for the stimulus to examine dynamic virtual organisations. This work has been partly supported by the EU FP7 Network of Excellence on Resilience in Information Society Technologies (ReSIST) and the EPSRC Platform Grant on Trustworthy Ambient Systems.

Bibliography

[1] J. W. Bryans and J. S. Fitzgerald. Formal Engineering of XAML Access Control Policies in VDM++. In M. Butler, M. G. Hinchey, and M. M. Larrondo-Petrie, editors, *Formal Methods and Software Engineering: Proc. ICFEM 2007*, pages 37–56. Springer-Verlag Lecture Notes in Computer Science 4789, 2007.

[2] J. W. Bryans, J. S. Fitzgerald, C. B. Jones, and I. Mozolevsky. Dimensions of Dynamic Coalitions. Technical Report CS-TR-963, Newcastle University, School of Computing Science, May 2006.

[3] J. W. Bryans, J. S. Fitzgerald, C. B. Jones, and I. Mozolevsky. Formal Modelling of Dynamic Coalitions, with an Application in Chemical Engineering. In T. Margaria, A. Philippou, and B. Steffen, editors, *IEEE-ISoLA 2006: Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, pages 90–97, November 2006.

[4] Steve Burbeck. Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html, 1978.

[5] J. S. Fitzgerald and P. G. Larsen. *Modelling Systems – Practical Tools and Techniques in Software Development*. Cambridge University Press, 1998. ISBN 0-521-62348-0.

[6] J. S. Fitzgerald and P. G. Larsen. Triumphs and Challenges for the Industrial Application of Model-Oriented Formal Methods. In T. Margaria, A. Philippou, and B. Steffen, editors, *Proc. 2nd Intl. Symp. on Leveraging Applications of Formal Methods, Verification and Validation*. IEEE, 2007. Also Technical Report CS-TR-999, School of Computing Science, Newcastle University.

[7] J. S. Fitzgerald, P. G. Larsen, P. Mukherjee, N. Plat, and M. Verhoef. *Validated Designs for Object-oriented Systems*. Springer Verlag, London, 2005. ISBN 1-85233-881-4.

[8] Peter Houghton. Potential System Vunerabilities of a Network Enabled Force. In *Proceedings of Coalition Command and Control in The Networked Era*, 2004.

[9] Trevor Nash. A Time to Refocus C4ISTAR Training. *RUSI Defence Systems*, 10(1):114–115, June 2007.

[10] Colin Robson. *Real world research : a resource for social scientists and practitioner-researchers*. Blackwell Publishers, Oxford, UK ; Madden, Mass., 2nd edition, 2002. Colin Robson. ill. ; 26 cm.

[11] Marcel Verhoef, Peter Gorm Larsen, and Jozef Hooman. Modeling and Validating Distributed Embedded Real-Time Systems with VDM++. In Jayadev Misra, Tobias Nipkow, and Emil Sekerinski, editors, *FM 2006: Formal Methods*, pages 147–162. Lecture Notes in Computer Science 4085, 2006.