Use of Roles and Policies for Specifying and Managing a Virtual Enterprise

Emil Lupu[§], Zoran Milošević[‡] and Morris Sloman[§]
[§]*Department of Computing, Imperial College, London, UK*
e.c.lupu, m.sloman@doc.ic.ac.uk
[‡]*Distributed Systems Technology Centre, University of Queensland, QLD 4072, Australia*
zoran@dstc.edu.au

Abstract

One of the problems faced by an organisation participating in a virtual enterprise is how to specify internal and external aspects of the organisation in terms of the organisational roles involved and policies applicable to the roles. Another problem is how to manage such a virtual organisation and guarantee that its operations are in agreement with the specification. In this paper we present two role-based approaches that address these problems. The first approach is based on the RM-ODP framework and can be used to specify structure and interactions in a virtual enterprise. The second approach was initially aimed at managing large distributed systems, but can also be used to specify and implement roles and policies relating to a virtual enterprise. We analyse the relationships between these two approaches and illustrate how they can be applied by means of a simplified virtual hospital example.

1. Introduction

A virtual enterprise is an organisation created from physically distributed constituents which are linked electronically to enable interaction and cooperation normally associated with a centralised enterprise, e.g., a virtual hospital, university, consultancy company or shopping mall. This linkage is established to achieve an overall objective such as enabling more flexible interactions between autonomous units, better joint competitive position or reducing costs associated with business interactions. This kind of enterprise is emerging owing to the capabilities of distributed object and component technologies and the increasing connectivity and bandwidth of the Internet.

Although these technologies provide a good base for establishing electronic interactions, it is necessary to be able to specify *policies* relating to the rights and duties of the interacting entities. This can be accomplished by defining *organisational roles* and relationships between them.

Prof. Sloman's group at Imperial College have developed a notation with associated tools for specifying, analysing and enforcing obligation and authorisation policies for managing large-scale distributed systems [1-5]. This includes a framework for the specification of management roles for distributed systems [6]. Although initially aimed at management, the framework can be used for specifying interactions and relationships between any roles within a virtual enterprise. Their concept of role corresponds to the policies defining rights and duties associated with an organisational position.

Recently, work has started in the ODP standardisation arena to produce a generic policy framework that would be able to address the problem of expressing obligations, permissions, prohibitions and a complex network of their relationships in open distributed systems. Some of the early ideas are presented in [7]. This work provides input to the current efforts of the ODP enterprise language standardisation [8], which will expand on the ODP notion of roles and their interactions, as well as the ODP concept of community

There are other efforts in which the problems of specifying policies and roles are addressed as part of enterprise modelling approaches. Examples are the concept of role from various object-oriented methodologies and/or notations, such as OORAM [9] and UML notation [10], the use of roles for representing transient object behaviour [25], and also some ideas coming from role-based access control systems [11].

In this paper, we concentrate on the Imperial College Role Framework (referred to as ICRF hereafter) and the ODP approach, and show how they can be used to address the problems mentioned above. The aim of the paper is to explore similarities and differences between these two approaches and investigate their use to address the problems faced by virtual enterprises.

Section 2 discusses the concept of role and related concepts as they are defined in the ISO/ITU-T Standards for Open Distributed Processing (ODP). Section 3 outlines the main ideas behind the Imperial College Role Framework. Section 4 provides comparative analysis of

these different concepts of role, discussing similarities and differences. We use a simple example of a virtual hospital to illustrate the concepts discussed throughout the paper. After a brief overview of the related work (section 5) conclusion and future research directions are outlined in section 6.

2. Role in ODP standards

2.1. ODP Definition of a role

Role is one of the foundation concepts in the ODP standards. It is defined as an "Identifier for a behaviour, which may appear as a parameter in a template for a composite object, and which is associated with one of the component objects of the composite object" [12].

This definition means that a role is a placeholder for behaviour to be filled by an object that satisfies this behaviour. A behaviour specification is "a collection of actions with a set of constraints on when they may occur" [12] associated with some object. There are many examples from real life which are concerned with describing interactions and relationships of a role such as ward-nurse, doctor, director within an enterprise, irrespective of which object (person) is filling the particular role.

In addition, the notion of role implies some context which explains how roles interact and the relationships between them, such as peer, sub-ordinate or super-ordinate roles. In fact, we are talking about the relationships between objects filling these roles, as these are actual instantiations of behaviour associated with these relationships.

The concept of role is extensively used within the ODP enterprise language and it represents one of the key concepts in the Enterprise Viewpoint [8]. A role type defines interaction behaviour but may include additional constraints on the behaviour, such as policy or Quality of Service (QoS) statements. Policy statements relate to the notions of obligations, permissions (authorisation) or delegation. QoS statements may describe requirements, capabilities, contracts in terms of error rates, throughputs, delay etc., relating to an interaction between roles.

2.2. ODP enterprise concept of community

An **ODP community** is defined as "a configuration of objects formed to meet an objective. The objective is expressed as a contract which specifies how the objective can be met" [13]. A *configuration* is a collection of objects, with defined relationships between them, able to interact at their interfaces. The *contract* is a set of objectives and constraints which govern the behaviour of the objects in the community.

A community type defines a set of community instances (including roles) whose behaviour is compliant with this description. An ODP role type specifies how the enterprise object, filling the role, behaves with respect to one or several other objects in the community. The enterprise objects can be IT-components, people or organisations. One enterprise object can fill more than one role in a community, and can be part of several communities.

A specification of a community type may define several role types, and each type may have several role instances. For example, a community instance for a small enterprise can have one instance (of an owner-role type) and say two instances (of an employee-role type). Now, an employee role instance can be filled by one enterprise object at a time, but the small enterprise community exists even if one (or both) of these roles are not filled. The assignment of enterprise objects to roles can take place when the community is instantiated or later e.g., when the work load justifies appointing a new employee to the post.

An enterprise specification may consist of several community specifications corresponding to multiple organisations. A role specified in one community can be referenced by a role specified in another community indicating dependency between different roles. A constraint may be that an object can satisfy one role only if it also satisfies some other role from the same or different communities. For example, a doctor can only be assigned as remote surgical consultant in Hospital A if he is a surgical consultant in the Hospitals X or Y. A role specification from one community can also be reused within another community specification, but this does not imply any relationship between community instances corresponding to these two community specifications. In general, one can state different constraints on objects that can fill roles in different communities, and these constraints represent policies of an enclosing community.

The community concept gives a specification of the virtual enterprise with the organisations involved, i.e. component communities. Roles define responsibilities, rights and duties of the actors or agents within the virtual enterprise.

2.3. An example: virtual hospital

We now describe some of the concepts by using a simple example of a virtual hospital in which many of the details are omitted for brevity.

A virtual hospital consists of several health-care institutions linked together by means of a high-speed network and through the use of an integrated software system deployed throughout these separate organisations. These health-care institutions can include hospitals, specialised clinics and community services. An informal

description of one part of a virtual hospital is shown in the Figure 1.

In this example, a consultant in a specialist clinic has an authorisation and obligation to prescribe a drug for a patient in the local hospital. This prescription is then stored in the local hospital database. The nurses assigned to the patient's ward are obliged to administer the prescribed drug at particular times. The head-of-ward has responsibility to check that the drug has been administered and the patient is responding to treatment.

When using the ODP enterprise language concepts, this scenario can be modelled as follows. The ward is a community, with roles of nurse, patient and head-of-ward. Each of the roles specifies behaviour of the enterprise object that may fill the role and the permission and obligation policies that apply to their behaviour. Here the behaviour is expressed in terms of actions and constraints on when they occur. For example, the specification of nurse role will include an action to administer the drug and at what time intervals. Typically, such an action will be constrained by the permission as to what kind of drugs this nurse can issue plus obligations to check the state of the patient before administering the drug and record what has been administered. Other authorisation policies limit what patient-information the nurse can access and obligation policies restrain the nurse from revealing some information to members of the patient's family.

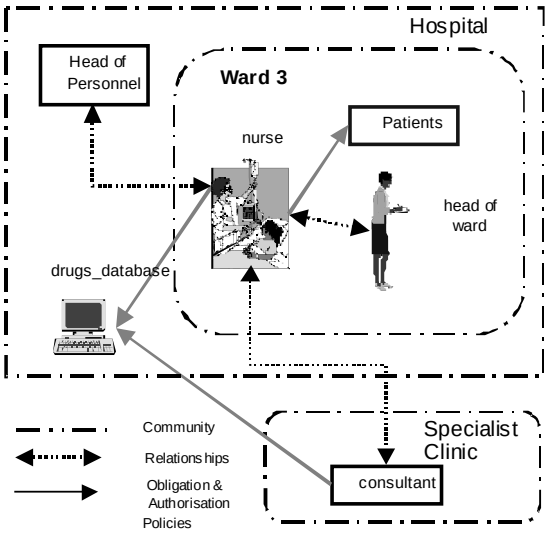

Figure 1 Virtual Co-operation in the health-care environment

The consultant role is specified in another community – that of a specialist clinic. Not all the details of this community are of interest for our specification. Rather, we will be concerned with the behaviour and the policies associated with the consultant role to an extent that it has implications on the ward community and the outer, hospital community of which this ward is part.

The consultant role type specifies action types such as a procedure for examination of patients, prescription of specific drugs and interactions with other medical professionals within (or beyond) the clinic or the hospital in the question. Typically, there will be constraints associated with this role such as preconditions on who can fulfil the role. For example, only those doctors who are certified (have permission granted to them) by an outer community (e.g., an association of immunologists) will be allowed to prescribe certain experimental drug for patients.

This is just a very small subset of the policies that would need to be specified for a virtual hospital. Many of the policies can be enforced by an underlying IT system. In the next section we show how some of these policies would be specified within the ICRF.

3. Imperial College Role Framework (ICRF)

The Role-Based framework was initially developed in order to provide the means of structuring delegated management in large distributed systems [1]. The framework includes a notation and tools for specifying obligation and authorisation policies defining the rights and duties of agents (actors) in the organisation [1]. Policy can be specified for groups (domains) of objects [2] and policy specifications may be grouped into roles to which organisational agents can be assigned or removed dynamically [6, 14]. Policies may be specified in multiple communities so the framework includes tools for analysing policies to detect conflicts [3].

3.1. Domains and Policies

Domains [2] are essentially a means of grouping objects and may contain agents, resources or other domains; thus defining a directed acyclic graph according to the inclusion relationship. They are used to partition the enterprise scope according to geographical boundaries, administrative departments, object type, etc. For example, objects inside a department, a project team or resources of a given type may be grouped in a domain. Domains are different from communities in that they specify a *group* rather than a *configuration* of objects. However, a domain, combined with the role relationships and policies of the ICRF can be used to model an ODP community and its associated role specifications. We have a domain service implementation but will be moving to a commonly available directory service with domains implemented as directories.

Policies [1] establish a relationship between the domains of agents and domains of objects which are targets of the agents' activities. A policy applying to a domain, propagates to all the objects in that domain including sub-domains and their members; thus making it

possible to specify policies for large numbers of objects in a hierarchical structure. Since policies are used in order to specify the rights and duties of the organisational agents, we distinguish between authorisation and obligation policies.

Authorisation policies define what activities a set of subjects (agents) can perform on a set of target objects e.g.

P1 A+ @/ward3/nurses { administer(analgesics) }
x:@/ward3/patients
when (x.temperature > 37) && (x.temperature < 38.5)

Nurses in ward 3 are authorised to administer analgesics to patients when their body temperature is between 37 and 38.5. Note the use of the constraint to limit the scope of applicability of the policy. The '@' indicates the policy propagates to all non-domain objects.

P2 A- @/ward3/nurses { validate() } @/ward3/patient_discharges

Nurses are not allowed to validate patient discharges

Obligation policies define what activities an agent must or must not perform on a set of target objects. Positive obligation policies are triggered by events and constraints can be specified to limit the applicability of the policy based on time or attributes of the objects to which the policy refers [4].

P3 O+ on too_high_temperature(x:patient) @/ward3/nurses {
administer(analgesics) } @/ward3/patients/x

This positive obligation policy is triggered by an event signalling that a patient's temperature is above a pre-set threshold and obliges the nurse to administer analgesics to the patient.

P4 A+ @/ward3/nurses { communicate(results) }
@/ward3/patients

Nurses are permitted to communicate test results to patients.

P5 O- @/ward3/nurses { communicate(results) }
@/ward3/patients
when results.overall = "critical-condition"

This negative obligation policy specifies that nurses must refrain from disclosing test results to patients when the results outline the patient's condition as critical, even though they are normally permitted to communicate test results. Note, that patients may want to know the results even though this may cause them a distressing condition.

P6 A+ n:@/ext-consultant { prescribe_experimental_drugs () }
@/ward3/patients
when n = immunoligist_certified

Only external consultants certified by the immunologists association can prescribe experimental drugs for patients in ward3.

The **subject** domain of a policy groups the human or automated agents to which the policies apply and which interpret obligation policies. The **target** domain of a policy groups the objects on which actions are to be performed. Security agents at a target's node interpret authorisation policies [5] and agents in the subject domain interpret obligation policies [4]. An advantage of specifying policy scope in terms of domains, is that objects can be added and removed from domains to which policies apply without having to change the policies. The **actions** e.g., administer(), validate() specify what must be performed for obligations and what is permitted for authorisations.

The above examples relate to concrete actions which can be performed by automated components, but many organisational policies are specified at a higher level of abstraction and progressively refined into more concrete policies. These specify actions which are either operations on the target objects or operations defined in the agent's code. For example, policies P1 and P3 above may be refined from a more abstract policy specifying that nurses are responsible for administering medication to patients, e.g.,

H1 O+ nurses { administer medication } patients

This refinement hierarchy is maintained by references from the abstract (parent) policies to the policies which have been refined from them. Furthermore, references to related policies can be maintained, e.g., from an obligation policy such as P3 to the authorisations policies permitting the actions (here P1).

3.2. Roles and Relationships

While policies may be used to specify the rights and duties for groups of agents in the organisation, they are also used to specify the behaviour expected from agents assigned to particular organisational positions. Therefore, we define roles as a set of policies applying to the same subject domain, called the *position domain* (Figure 2). Agents can then be assigned to or removed from a role without re-specifying the role's policies.

Roles interact with each other and have duties and rights towards each other. For example, a nurse has the duty to report to the head-of-ward, which is responsible for assigning new tasks to the nurse. Furthermore, the nurse must request permission from the head-of-ward before moving a patient to a new bed to which the response may be either an agreement, a denial or a referral to the treating consultant. Additional interaction protocols for collaboration between the two roles may exist in order to coordinate their actions in case of emergency. In the ICRF framework we have therefore introduced the concept of *relationships* which group the rights and duties (i.e., the obligation and authorisation policies) of the

related roles towards each other and the interaction protocols which regulate the exchanges of messages between them. Additional elements of the framework including structural and concurrency constraints are described in [14].

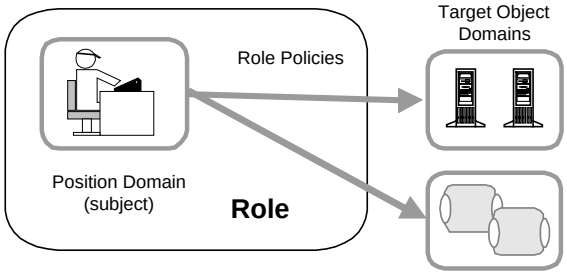

Figure 2 Roles

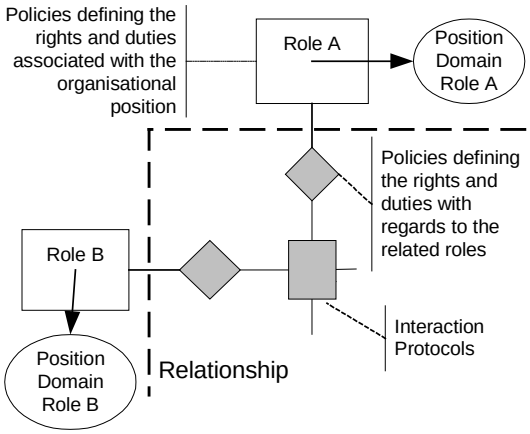

Figure 3 Roles and Relationships

Figure 3 illustrates some of the concepts described above. Roles contain the policies influencing the behaviour of the agents assigned to them. These policies have a common subject domain: the position domain. Furthermore, roles participate in relationships which define the policies regarding their behaviour towards the related roles and the interaction protocols which constrain the exchanges of messages within the relationships. The interaction protocol definition is based on the specification of production rules constraining the possible responses on receipt of a message. This approach to interaction protocol specification [6, 14] is similar to the approach adopted in [15] although there are substantial differences in the production rule notation and interaction semantics.

In the case of the virtual hospital described in Figure 1, the responsibility of the consultant to examine and prescribe medication to the patients would be specified as policies which are part of external-consultant role, while obligations to provide details to the nurse regarding how drugs must be administered and interactions with the nurse to deal with emergency situations would be specified as part of the relationship between the external-consultant and the nurse.

3.3. Role and Relationship Classes

An organisation may contain large numbers of roles with few differences between them. Furthermore, each role may be part of a large number of relationships. We therefore introduce classes and templates in order to reduce the number and complexity of the specifications. For example, a nurse role class can be specified and used to create the nurse-instance roles for wards 3,4 and 10. Each instance may then be customised for any particular task relating to a specific ward and a specific person assigned to each role. The definition of role classes is based upon policy templates (which are specifications of rights and duties independent of subject, target or both). Just as a role groups a set of policies, a role class groups a set of policy templates for which subject and/or target are specified when creating an instance from the class. In particular the subject is defined by associating the role with a position (i.e., a Position Domain).

Role classes specify the policy templates defining the rights and duties of a generic role in the organisation e.g., nurse, engineer, marketing manager. Instances are then created from the classes in the various domains of the organisation, e.g., nurses in each ward of the hospital.

Single and multiple inheritance can be defined between role classes in order to implement specialisation and re-use of the specifications (e.g., Figure 4).

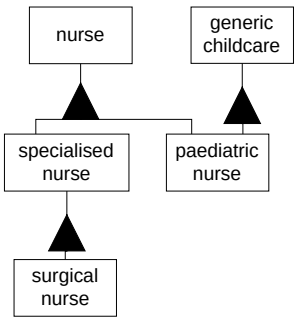

Figure 4 Role Class Inheritance

Relationship classes can also be defined by parameterising the participants in the relationships. A relationship class therefore defines the policies and interaction constraints common to a set of participants, which will be specified when an instance is created. Note, that some of the participants can be specified in a relationship class. This has as effect that all the instances created from that class will be relationships between the roles (specified in the class) and the roles which are defined during the instantiation. For example, in the virtual hospital described in Figure 1 the head of

personnel is responsible for reviewing the salary of each nurse at the end of the year and handling the review for promotions and bonus payments. This can be specified in a relationship class which contains the head-of-personnel role instance and another participant (i.e., the nurse) which will be specified when the class is instantiated.

Single inheritance can be defined between relationship classes in order to specialise and re-use the relationship specifications.

A relationship class must specify the type (i.e., the role class) for each of the participants in order to verify, during instantiation, that the roles bound in the relationship can fulfil the policy and interaction requirements specified in the relationship. Furthermore, role classes can refer to relationship classes therefore specifying that a role instance cannot be created without associating it in a relationship of the given type. For example, a reference from the nurse role-class to the nurse/head-of-ward relationship would indicate that a nurse role-instance cannot be created without relating it to a head-of-ward. These requirements place constraints on the instantiation process which amount to defining organisational patterns. The configuration of role and relationship classes which includes the head-of-ward, nurses and the consultant can be defined for a ward of a hospital. This configuration can then be instantiated for each ward of the hospital. Decisions regarding the number of nurses or which nurses assist which consultants are must be made during the creation of the instances, although these decisions may be subject to structural and cardinality constraints.

While most of the examples here relate to the health-care environment other case-studies regarding software development teams and administration and maintenance of cellular networks have been specified in the prototype framework developed at Imperial College.

4. Discussion

The ODP enterprise concepts and their manifestation within the ICRF provide a means of specifying the policies relating to roles within the constituent organisations within a virtual enterprise. This type of specification is essential to define what potential interactions are permitted and the obligations pertaining to the roles. This specification is likely to be stable over periods of days to months. It would have to be produced manually by negotiation between administrators within the constituent organisation or could be specified by an enclosing community e.g., a provincial or town health authority for the virtual hospital example.

The objects fulfilling roles can be much more dynamic than the community specification and a doctor may fulfil a consultant role for only a few minutes, or a nurse may be assigned to a ward for a few hours. The maximum number of instances of a role, such as nurse, could be predefined for a ward but for some applications it may be necessary to create new role instances dynamically. It is also possible to dynamically change, enable and disable policies and to dynamically modify the membership of domains.

Some policies will be specified independently by administrators in each of the organisations within a virtual enterprise. Since several policies may apply to an object, conflicts may arise between them. For example, the policies of the consultant's home hospital may prevent consultants from directly assigning new duties to nurses, while this may be required of him in another hospital. It is thus necessary to detect and resolve policy conflicts. This is presented in [3] and is beyond the scope of this paper.

There are a number of common concepts between the ODP Enterprise viewpoint and the ICRF. A means of grouping objects is essential. We have indicated that the ODP *community* can be mapped into the ICRF *domain* with role and relationships defining the interaction between roles and global policies specifying overall constraints on the domain.

Both frameworks identify the concept of *policy* in order to explicitly attribute responsibility to agents in the organisation. Policies are essentially of two kinds: obligations and authorisations, although other studies also introduce the concept of freedom policies, e.g., [16]. The ODP Enterprise Language does not yet have a clearly defined notation for specifying policies, although they express policies in terms of deontic statements (obligation, permission and prohibition) as defined in [12]. The ICRF introduces the concept of negative obligation policy (not found in Deontic Logic) in order to distinguish between prohibitions i.e., actions for which the security sub-system must deny access to the target objects, and actions which the subjects themselves 'must refrain' from performing. Since prohibitions (i.e., negative authorisations) are specified in order to protect the target objects from unauthorised access by subjects, the corresponding policies must be interpreted by trusted access control agents on the *target* system [5]. Negative obligations are necessary in those situations where target objects cannot be relied upon in order to protect themselves from unauthorised access. For example in policy P5 (Section 3.1) the patients may desire to know the results of their medical examinations no matter what condition they reveal. In addition, nurses may actually be authorised to communicate to patients the results of their medical examinations. A similar use of negative obligation policies may also be found in [17].

Roles are used in both approaches to group the specification of the behaviour which is expected from the objects assigned to them. The behaviour is specified either as policies which are part of the role and its associated relationships (ICRF framework) or as policies specified in the *contract* of the community in which the role is

included (ODP). In both cases roles are used as a placeholder to enable agents to be dynamically assigned or removed without changing the role specifications. This assignment may be subject to additional constraints and a type compatibility check. Although the ODP Enterprise Viewpoint makes provision for specifying relationships between the roles of a community, work on this topic is still in progress. Nevertheless, both the Enterprise Viewpoint and ICRF specify the rights and duties of the related parties towards each other and the interactions between them. This enables specification of many types of relationships of different arity and nature, e.g., contractual relationships, producer consumer relationships, functional hierarchies [18], etc.

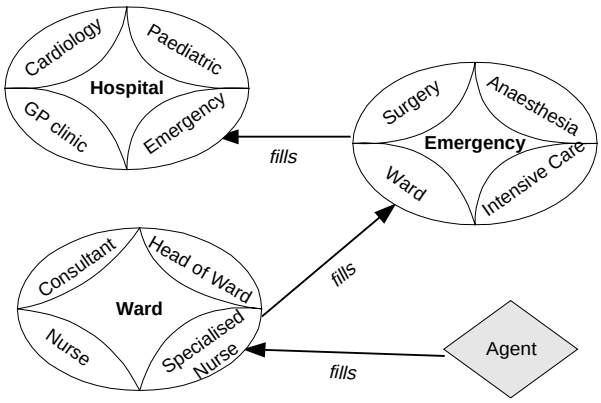

Figure 5 Use of Communities with composite enterprise objects

While in ICRF domains are the only means of grouping objects, the ODP Enterprise Viewpoint also uses composite objects in order to represent components of an organisation e.g., departments, teams or an entire organisation. Each of these enterprise objects could then be assigned to a role in another community. Thus, a hospital could be represented as a top-level community where each of the departments are playing a particular role e.g., the Cardiology department, Emergencies, Paediatric Care or GP clinic (Figure 5). The Emergency role in the Hospital community is filled by the Emergencies department (viewed as enterprise object), which is itself a community composed of several roles. Similarly, the Ward role in the Emergency community (Figure 5) is filled by the Ward community, which is an enterprise object. ICRF has derived the notion of roles from classical role-theory [19] where roles relate to positions within an organisation.

5. Related Work

The concepts of policies and roles occur in many different areas. Role Based Access Control, used for security [11] does not cater for the specification of obligations and adopts a different approach to role inheritance [20]. Object-Oriented Modelling Frameworks [9,10] define role in a way similar to the ODP concept, as they consider roles as first class modelling elements in cases where the focus is on behaviour of an object with respect to interactions with other objects. The concept of association describes roles and their interactions. It is not clear how one can go about specifying policies associated with the roles and how one can describe relationships between roles belonging to different associations.

Roles have also been introduced in Object-Oriented Databases as an extension to classes in order to support dynamic changes of behaviour of an object or a group of objects while maintaining the object identity [25]. A long-lived object can therefore migrate between roles at run-time without requiring its re-classification, i.e., deletion and instantiation from a new class. This perspective differs substantially from the ICRF in that behaviour is assumed to be executable content (attributes and methods) associated with objects rather than interpreted policies which define the authorisations and obligations of dedicated agents. Obligation policies define what actions must be performed and when, not what the actions consist of.

The Role Interaction Nets [21] provide a formalism based on roles, teams and processes for specifying and building distributed interactions in an organisational setting. Their framework is based on the concept of n-party synchronous interaction [22].

In telecommunication service architectures roles have recently be introduced in order to model relationships between service providers and subscribers or between the service providers themselves. However, tele-communication service roles tend to be more transient in nature or even dynamically generated on a per-session basis [23].

6. Conclusions and Future Work

This paper investigates the applicability of two enterprise modelling approaches for specifying, building and managing virtual enterprise. These two approaches are overlapping in that they address a similar set of concerns viz the specification of policies, roles and their relationships. They are also complementary in that the ICRF is more pragmatic and, owing to the notation provided and tools developed, applicable for the implementation, monitoring and enforcing policies associated with organisational roles. The ODP approach is still in an early stage with no clearly defined language and hence no ODP conformant tools.

This paper represents our first step in establishing similarities and differences between the concepts of these two approaches. Our next step is to further relate these two approaches and study in detail some specific

enterprise modelling issues as follows. First, we plan to extend both of these frameworks to better deal with different kinds of relationships between ODP communities. Second, we are interested in gaining a better understanding of the concept of organisational objective and how it can be further refined in temporal, actions and non-functional spaces. Some of the results from the Requirements Engineering Community [24] may be exploited. Finally, we intend to extend the study of the roles, policies and related concepts by a comparative analysis with similar concepts in various O-O methodologies. This can serve two purposes: enriching O-O methodologies such as UML with the necessary concepts for Enterprise Modelling and also using UML's associated graphical tools to produce Enterprise Viewpoint specifications.

7. Acknowledgements

We gratefully acknowledge financial support from the EPSRC for a Visiting Fellowship grant number GR/M 15804, from Fujitsu Network Systems Laboratories for the Pro-Active Role Based Management for Distributed Services project and from British Telecom for the Management of Multimedia Networks project. The work has also been funded in part by the Co-operative Research Centre Program through the Department of Industry, Science & Tourism, Australia.

8. References

[1] M. Sloman "Policy Driven Management for Distributed Systems". *Plenum Press Journal of Network and Systems Management*, 2(4):333-360, 1994.

[2] M. Sloman and K. Twidle, "Domains: a Framework for Structuring Management Policy". Chapter 16 in *Network and Distributed Systems Management*, M. Sloman ed., Addison Wesley, 1994, pp. 433-453.

[3] E. Lupu and M. Sloman, "Conflicts in Policy-Based Distributed Systems Management". To appear in *IEEE Transactions on Software Engineering* - Special Issue on Inconsistency Management, 1999.

[4] D. Marriott and M. Sloman, "Implementation of a Management Agent for Interpreting Obligation Policy", In Proc. IEEE/IFIP Distributed Systems Operations and Management Workshop, L'Aquila, Italy, 1996.

[5] N. Yialelis and M. Sloman, "A Security Framework Supporting Domain-Based Access Control in Distributed Systems". In Proc. IEEE/ISOC Symposium on Network and Distributed Systems Security, San Diego, pp. 26-34, Feb. 1996.

[6] E. Lupu and M. Sloman, "Towards a Role-Based Framework for Distributed Systems Management". *Plenum Press Journal of Network and Systems Management*, 5(1):5-30, 1997.

[7] P. Linington, Z. Milosevic and K. Raymond, "Policies in Communities: Extending the ODP Enterprise Viewpoint". Proc. 2nd IEEE Enterprise Distributed Object Computing Workshop, San-Diego, Nov.1998.

[8] ISO/IEC JTC1/SC33/WG7, "Information Technology - Open Distributed Processing - Reference Model - Enterprise Viewpoint," ISO ISO/IEC 15414 | ITU-T Recommendation X.9ff, Jan. 1998.

[9] Numerica Taskon, "OOram Professional", 1998, http://www.sol.no/numerica/ooram.htm

[10] Unified Modelling Language version 1.1, Rational Software Corporation. Sept. 1997. Available from http://www.rational.com/

[11] Proc. of the 2nd ACM Workshop on Role Based Access Control. Fairfax, Virginia, USA. ACM Press, Nov. 1997.

[12] ISO/IEC JTC1/SC21, "Basic reference model of open distributed processing, part 2: Descriptive model," ITU-T X.902 - ISO/IEC 10746-2, Aug. 1994.

[13] ISO/IEC JTC1/SC21, "Basic reference model of open distributed processing, part 3: Architecture," ITU-T X.903 - ISO/IEC 10746-3, 1995.

[14] E. Lupu and M. Sloman, "A Policy Based Role Object Model". In Proc. 1st IEEE Enterprise Distributed Object Computing Workshop, Gold Coast, Australia, pp. 36-47, Oct. 1997.

[15] N.H. Minsky and V. Ungureanu "Regulated Coordination in Open Distributed Systems". In Proc. 2nd Int. Conf. on Coordination Models and Languages, Berlin, Germany, Sept. 1997.

[16] D. Jonscher, "Extending Access Control with Duties, realized by active mechanisms". In Proc. IFIP WG 11.3 6th Working Conference on Database Security, Vancouver, Aug. 1992.

[17] N. H. Minsky and A. D. Lockman, "Ensuring Integrity by Adding Obligations to Privileges". In Proc. 8th Int. Conf. on Software Engineering, London, U.K., pp. 92-102, Aug. 1985.

[18] T. W. Malone, and K. Crowston. "The Interdisciplinary Study of Coordination". *ACM Computing Surveys*, 26(1):87-119, May 1994.

[19] B. J. Biddle, and E. J. Thomas, *Role Theory: Concepts and Research*. Robert E. Krieger Publishing Company, Huntington, New York, 1979.

[20] E. C. Lupu and M. S. Sloman, "Reconciling Role Based Management and Role Based Access Control". In [12], pp. 135-142.

[21] B. Singh and G. L. Rein. "Role Interaction Nets (RINs): A Process Description Formalism". Technical Report No. CT-083-92, Microelectronics and Computer Technology Corporation (MCC), Austin, Texas, USA, July 1992.

[22] I. R. Forman, "On the Design of Large, Distributed Systems". Technical Report No. STP-098-86, Microelectronics and Computer Technology Corporation (MCC), Austin, Texas, USA, March 1986.

[23] T. Hamada "Dynamic Role Creation from Role Hierarchy". In Proceedings of TINA'97, Santiago, Chile, Nov. 1997.

[24] A. van Lamsweerde, R Darimont and P. Massonet. "Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt ". In Proc. 2nd IEEE Int. Symposium on Requirements Engineering, York, U.K., pp. 194-203, March 1995.

[25] M. P. Papazoglou and B. J. Kramer. "Representing Transient Object Behavior". *VLDB Journal* 6(2):73-96, May 1997.