

# Dynamic Policy Model for Large Evolving Enterprises

Nicole Dunlop †§, Jadwiga Indulska †§, Kerry Raymond §

† School of Computer Science and Electrical Engineering,  
The University of Queensland, Australia

§ CRC for Enterprise Distributed Systems Technology (DSTC), Australia

E-mail: {dunlop, jaga}@csee.uq.edu.au, kerry@dstc.edu.au

## Abstract

*The nature of an open distributed environment provides a resoundingly diverse yet potentially chaotic environment for users. A great deal of research has focused on the management of resources in such an environment and policy-based management has emerged as one such promising solution.*

*In order to support large evolving enterprises we have developed a policy model that is scalable, able to cope with the extraordinarily high rate of change inherent in such environments and recognises the efficiencies to be gained from supporting trust within enterprise communities. To do this we have developed a number of novel concepts including, enterprise domain, policy space, policy authority, compound actions and defences which are unique and central to our approach.*

*Furthermore, our model supports dynamic roles, dynamic policies and dynamic conflict analysis. Prevailing policy-based management models largely promote the static definition and analysis of policy. We argue that although these models are suitable for homogenous, largely static environments, they are too rigid to adequately represent large, heterogeneous, evolving enterprise as the fluidity and complexity of interactions occurring in such environments are genuinely difficult and often impossible to pre-empt at the time of policy specification and role assignment.*

## 1 Introduction and Motivation

It is well understood that the increasing importance of open distributed systems and the growing number of services that utilise them has given rise to the need for effective distributed systems management [24]. Furthermore, it has also become increasingly difficult to build management systems that can cope with the increasing size and complexity of current open distributed systems [25, 27].

The policy concept has been gaining wide acceptance as a means for enforcing enterprise-specific procedures; however, currently there is no clear consensus within the research community on what exactly defines a *policy*. The Oxford English Dictionary defines policy as a “*method or principle of action adopted or proposed by an authorised body*” [20], the American Heritage Dictionary defines policy as “*a general principle or plan that guides the actions taken by a person or group*” [18]. We believe that from the enterprise perspective, policies can be defined as a “*course of action determined by an authorised entity for the purpose of constraining the behaviour of communities in such a way as to achieve the common enterprise objectives*”.

The central premise of the policy-based management model is that users do *not* have discretionary access to enterprise objects; instead permission to access enterprise objects and duty to perform assigned organisational obligations (*policy*) are associated with roles. Users can then be assigned to roles as determined by their responsibilities and qualifications.

The value of the policy-based management model lies in its simplification of both the specification and enforcement of enterprise-specific policies. Users are not assigned policy to perform operations on an individual basis, but rather policies are associated with roles, of which, users become members. As the enterprise evolves, role association with new and modified operations can be established as well as redundant operations deleted as organisational functions change. This fundamental concept has the advantage of simplifying the management of enterprise-level policy (widely acknowledged as an arduous process involving considerable recurring expense) [7], without the need to modify the underlying access infrastructure.

To make evident the importance of managing enterprise-level policies and the significance of policy-based models as a means of achieving this, consider the outrageous, unconstrained actions of Nicholas Leeson, whose fraudulent and reckless trading on the futures exchange resulted in a

£1.3 billion loss and the irretrievable financial collapse of the worlds oldest bank, Barings PLC [1, 6]. Barings PLC, valued at £500 million was sold the next day for £1 sterling. The collapse of the Barings enterprise can be directly attributed to the mismanagement of the time-honoured principle of *separation of duty*. Leeson was able to run both the financial derivatives trading operation as well as the back-office functions where trades were settled, management at Barings PLC should have recognised the conflict of interest inherent in allowing the same person to make and settle trades and implemented effective management procedures to ensure against it. Such a conflict of interest can easily be specified and enforced using an appropriate policy-based management model.

Traditionally, policy-based management models have considered that roles, their membership and the policies assigned to them are able to be determined *statically* in advance by the application designer. A role in this context is a *consistently specified* collection of policies governing the actions of a *consistently specified* collection of assigned users. While such models would be useful in a variety of essentially unvarying application environments they require *anticipation* of potential behaviours, events and their likely outcomes and therefore, lack the flexibility and responsiveness required of complex, open distributed enterprises.

Open distributed enterprises commonly exhibit certain key characteristics, which are scalability, rate of change and trust as defined below:

- **Scalability.** Due to the continually varying nature of the market within which enterprise operates, it is imperative that systems be scalable. Scalability refers to the ease with which a system can adapt to and accommodate increasingly complex requirements without the need for substantial changes to system structure or application algorithms.

Some factors which have a direct impact on the level of scalability within distributed enterprise systems are, increasing user base, evolving and escalating functionality requirements, emerging distributed computing domains and heterogeneity of distributed computing domains.

- **Rate of Change.** Evolving enterprises must be able to cope with the extraordinarily high rate of change ubiquitous in such environments. Including, merging distributed computing domains which often results in the introduction of new services and a need for heterogeneous applications to co-operate.
- **Trust.** Trust is essential for the efficient operation of large, evolving enterprises. Communities within an enterprise frequently recognise common organisational objectives and the members of those commu-

nities must typically exhibit a level of qualification, experience and responsibility in order to be included within the community. It is therefore reasonable that we allow those members to operate with a degree of independence deemed appropriate to their level of qualification and the nature of the actions they are performing in order to ensure efficient interactions within the enterprise.

By ensuring support of the above key characteristics of open distributed enterprise, we deliver a more flexible model of policy-based management suited to large, evolving enterprises.

This paper begins by providing descriptions of the basic concepts in our dynamic policy-based management model in Section 2, followed a discussion of the use of hierarchies in our model to support scalability in Section 3. Section 4 describes our use of dynamism to support system evolution and Section 5 discusses our use of optimistic measures to support trust in open distributed environments. In finalising, Section 6 will discuss the related work in the area of policy-based management and Section 7 is the conclusion and discussion of further work.

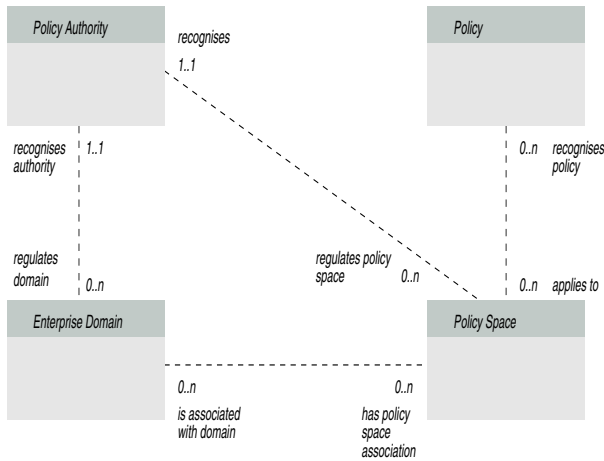
## 2 Basic Concepts in our Model

The core components of our dynamic policy model, as represented in the UML [19] diagram, Figure 1, (a partial representation of our complete policy model), are *policy*, *policy authority*, *enterprise domain* and *policy space*. In our model, actions and entities are grouped together in *policy spaces* and *policy* is then written to apply to the actions and entities belonging the *policy space*. The *policy* is then written and subject to the regulation of the assigned *policy authority* and are commonly associated with *roles* that are contained within the *policy space*.

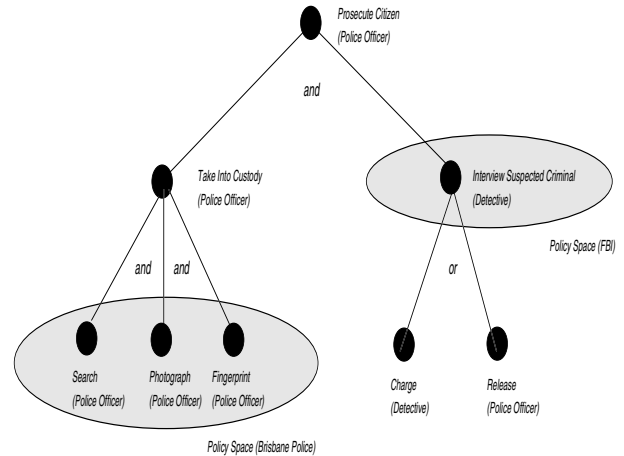
### 2.1 Action Concept

An *action* is the process of performing an operation, it represents the task or application being run. In current policy-based management models, actions are atomic. We argue that this is an inaccurate observation of reality, where actions are commonly more complex (compound), see Figure 2. Furthermore, the user initiating the action may not, in fact, be capable of performing all associated sub-actions.

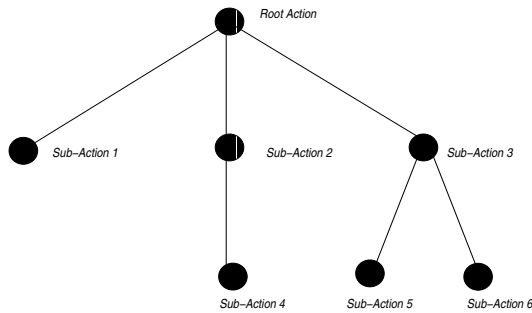
For example, in Figure 3, the action *Prosecute Citizen* cannot be completely fulfilled by the initiator of the action,



**Figure 1. Core High-Level Components of the Dynamic Policy Model.**



**Figure 3. Example of a Compound Action.**



**Figure 2. Compound Actions.**

**police officer.** The sub-action of *Interview Suspected Criminal*, for example, is a sub-action that can only be fulfilled by a user acting in the role of **detective**, whereas the root action to *Prosecute Citizen* was an action instigated by a user acting in the role of **police officer**.

Furthermore, sub-actions may be conditional. For example, in Figure 3 the actions of *Search*, *Photograph* and *Fingerprint* must necessarily all be completed, whereas the actions of *Charge* or *Release* are conditional upon the results of the action *Interview Suspected Criminal*.

## 2.2 Entity Concept

An *entity* is an object that we are interested in writing policy about. Entities may be Users, Roles, Artefacts or Organisations. Mechanisms for assigning policy to roles are used, complemented by mechanisms for assigning users

or artefacts to roles. Policy is then assigned on the basis of the *roles* that a user or artefact is a member of, where a role can be considered to be a set of rights and duties usually associated with a particular position within an organisation, or a particular working activity. Thus, instead of specifying access rights and obligations in terms of individual users or artefacts, users or artefacts are simply assigned to roles as appropriate and traditionally comply with the authorisations and obligations of the roles to which they belong.

## 2.3 Policy Concept

The purpose of *policy* is to define or constrain the current or future behaviour of a person or group to ensure that their actions are aligned with the objectives of the enterprise.

Initially, policies are specified at an enterprise-level and represent statements about the organisations requirements and goals, see Figure 4. These statements are usually specified in some abstract natural language (eg. structured english) and are then refined into more concrete statements about organisational behaviour or *policy*.

Policy is widely considered in the literature to define *permissions*, *prohibitions* and *obligations* [2, 3, 4, 14, 21], and can be described as follows:

- *Permission* is “the action of permitting or giving leave; allowance; liberty or a licence granted to do something” [23].
- *Prohibition* is “an unambiguous statement or rule, regulating *prohibited* behaviour in a system” [20].
- *Obligation* is “an agreement, enforceable by law, whereby a person or persons become bound to a particular action or performance of some duty by a contract containing such an agreement” [20].

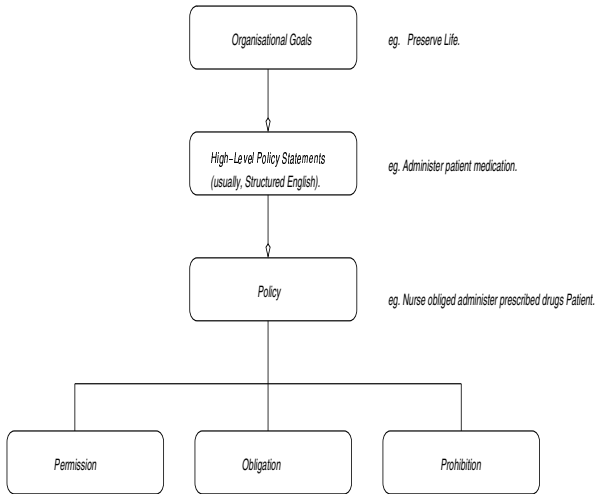


Figure 4. Policy Refinement.

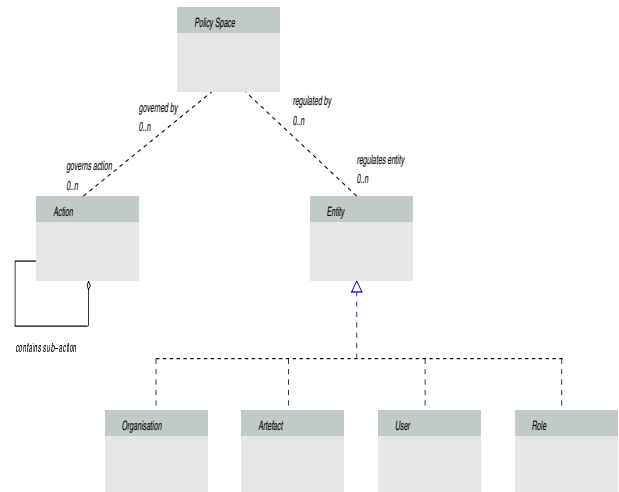


Figure 6. Representation of Policy Spaces.

## 2.4 Enterprise Domain Concept

An *enterprise domain* is an enterprise-level classification of all the entities within the organisation. As such, it is typically, but not always, a hierarchical division of departments, faculties, teams etc and is a convenient means by which we can *enumerate* objects under common authority.

## 2.5 Policy Space Concept

The purpose of identifying a *policy space* is to combine all of the entities and actions about which we **are interested in writing and applying policy**. A policy space is generally a declarative statement or description of the entities and actions about which policy will be written. For example, in Figure 5, policy can be written about *University Parents*, who will not be *obliged* to work past 6.30pm, due to child-care responsibilities. The concept of *policy space* is represented in Figure 6.

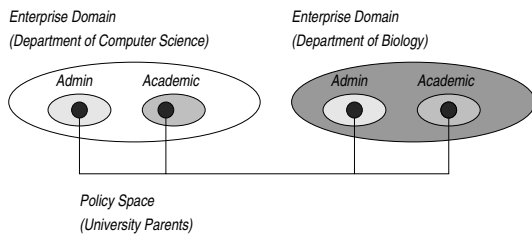


Figure 5. Example of Policy Spaces.

## 2.6 Policy Authority Concept

In our model, *policy authorities* are assigned to both Enterprise Domains and Policy Spaces. An Enterprise Domain Authority is the assigned owner of resources within that Enterprise Domain, whereas a Policy Space Authority is one which may write policy regarding the entities and actions contained within the Policy Space, see Figure 1. The concept of *policy authority* is clearly useful in the situation where applicable policies are in conflict.

## 3 Hierarchies to Support Scalability

In order to support scalable enterprise management we have developed a policy-based management model that extends the definition of *policy domain* from one which is defined in accepted literature as an “*hierarchical grouping of objects within a system for the purpose of collectively specifying management policy*” [3, 26] to one more supportive of co-operation between distributed computing domains, through the introduction of *enterprise domains* and *policy spaces*.

In our model, *enterprise domains* may be independent of each other, co-operating (as peers) or dependent on a senior domain, see Section 3.1. We believe this more realistically reflects the type of relationships domains typically have with each other in large, open distributed enterprise.

In addition we also define the concept of a *policy space*. A policy space is a collection of users and entities to which we are interested in applying policy. We define a *policy space* as distinct from an *enterprise domain* as the grouping

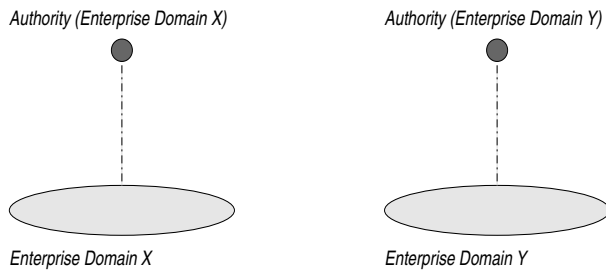
of actions and entities to which we would like to apply policy will frequently *span* enterprise domain hierarchies and will not always reflect the natural grouping of actions and entities as determined by the enterprise domain structure, see Section 3.2.

Finally, in support of scalable enterprise management, we establish the notion of *compound actions*. Compound actions are a recognition of the fact that actions are not simply references to atomic operations, but are commonly comprised of several related sub-actions, which may themselves belong to disparate policy spaces under the control of different authorities, see Section 2.1. Compound actions are commonly represented as a hierarchy of related sub-actions.

### 3.1 Enterprise Domain

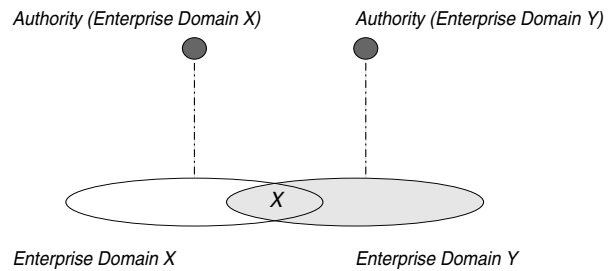
Enterprise Domains can be one of several types, described as follows:

- The top-level or *root domain* is used as an absolute reference point for the domain hierarchy.
- An *independent domain* is one that is subject to independent seniority, but is recognised as an associate of the domain specified by the *root*. Independent Enterprise Domains may be disjoint of each other, see Figure 7, or they may share common entities, see Figure 8.

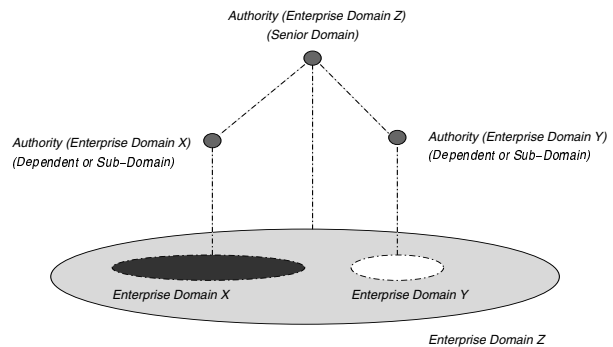


**Figure 7. Independent Enterprise Domain Types - No Shared Entities.**

- A *senior domain* is one which is recognised as the superior domain of a specified sub-domain or *dependent domain*, see Figure 9.
- *Dependent domains* or sub-domains are those subject to the policies of its *senior domain*, see Figure 9.

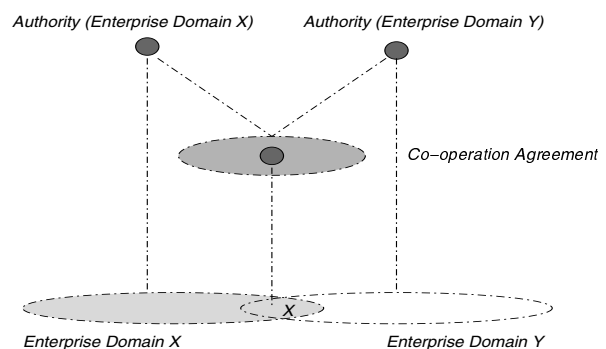


**Figure 8. Independent Enterprise Domain Types - Shared Entities.**



**Figure 9. Senior/Dependent Enterprise Domain Types.**

- domains may also be *co-operating*. Co-operating domains do not recognise seniority in each other, but have vested interest the other party in order to perform some business function, see Figure 10 .

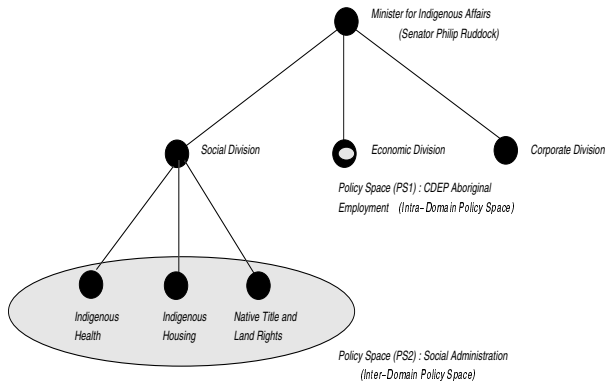


**Figure 10. Co-operating Enterprise Domain Types.**

This classification provides an elegant division of actions and entities which not only aids in the management of open distributed systems, but is also pivotal to the analysis and resolution of policy and role conflict.

### 3.2 Policy Space

A *policy space* may contain actions and entities that belong to a single Enterprise Domain *intra-domain* or may contain actions and entities that belong to many Enterprise Domains *inter-domain*. For example, Figure 11 contains the intra-domain Policy Space [PS1] and the inter-domain Policy Space [PS2]. All Policy Spaces will have a single assigned Policy Space Authority. Only the Policy Space Authority is able to write policy related to the entities and actions within the Policy Space.



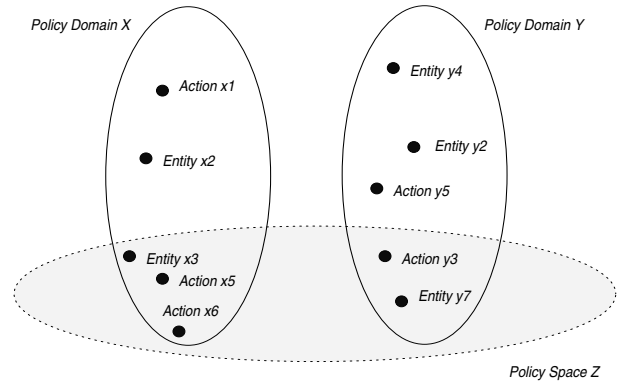
**Figure 11. Representation of Intra-Domain and Inter-Domain Policy Spaces within the Australian Department of Indigenous Affairs.**

### 3.3 Policy Authority

In the case of *intra-domain* Policy Spaces, for example, [PS1] in Figure 11, the Enterprise Domain Authority becomes the Authority of the Policy Space. *Inter-domain* Policy Spaces, for example, [PS2] in Figure 11, however, are somewhat more complex in that these Policy Spaces combine entities and actions belonging to a number of separate Policy Domains under control of possibly more than one Enterprise Domain Authority.

To further illustrate this idea, take for example, Figure 12, if the Enterprise Domain Authority (X)  $\equiv$  Enterprise Domain Authority (Y) it follows that there is one clear choice for the Policy Space Authority (Z). If, however, the Enterprise Domain Authority (X)  $\neq$  Enterprise Domain

Authority (Y), then the Policy Space Authority would generally be negotiated as a federated agreement between the participating Policy Domain Authorities (usually involving an examination of each Authorities position in the Enterprise Domain hierarchy), see Section 3.1.



**Figure 12. Representation of Inter-Domain Policy Spaces.**

### 3.4 Role Profile

In current literature [2, 4, 7, 13, 21, 26], a user is assigned a set of roles, however, it is unclear as to how the user relates to these roles. In our model we associate each user with a *role profile*. A *role profile* describes how roles are related for each individual user and is generally hierarchical in composition. This organisation of roles clearly shows which roles are more senior for this particular individual and which roles will therefore take precedence in the event of conflict. The *role profile* classifies roles in the same way as we classify *enterprise domains*, see Section 3.1.

## 4 Dynamism to Support Evolution

In order to support an evolving enterprise it is necessary to allow users to adapt to changing circumstances and organisational responsibilities and to make use of new services as they are introduced. Our model will allow users to acquire roles both at compile-time, *static roles* and at run-time, *dynamic roles*, see Section 4.1 and under certain circumstances allow users to more flexibly relate to their assigned policies at run-time, *dynamic policies*, see Section 4.2.

To this end, we define a dynamic policy-based management model aimed at supporting the following:

- Dynamic Policy Definition, allowing applications to interwork with both existing services and with new services as they are introduced (co-operation of applications). This will be realised through the support of policy that contains reference to run-time environment and through allowing users to dynamically modify policy under certain circumstances.
- Dynamic Role Definition, allowing users to adapt to changing circumstances and organisational responsibilities by enabling them to dynamically modify their role memberships.
- Dynamic Conflict Analysis, involves a process of both *conflict detection* and *conflict resolution*. Dynamic conflict analysis is required when policies or roles are introduced or modified (as a result of allowing applications to interwork with discovered resources and users to adapt to changing responsibilities and unforeseen circumstances).

Run-time conflict detection will determine if the new policy or role has caused a conflict with existing policies or roles and conflict resolution are defined techniques which will be used to determine a course of action in the event of conflict detection.

Role and policy-based management implies support for dynamic roles, dynamic policies and consequently, the dynamic conflict analysis of both policies and roles. This section characterises requirements for dynamic roles, policies and conflict analysis.

#### 4.1 Dynamic Role Support

As mentioned, roles may be assigned either statically (at compile-time) or acquired dynamically (at run-time). We have seen that static roles are sufficient in many circumstances as they provide a conceptual model for associating policies to users that can be efficiently implemented on top of common access control mechanisms [4]. However, traditional static roles do have limitations, as commented on in the literature [17],

*“There is a potential problem in systems which support the definition of social roles: “premature” definitions of these roles could lead to undesirable consequences.*

*For example, it is not always clear at the outset of a project who is going to make a “significant contribution” and therefore who should get [the] authorship [role]. But if authorship is defined at the outset, then it may reduce the motivation of someone who has been defined as a “non-author” and the person may not contribute as much.”*

Supporting dynamic roles recognises that organisational circumstances are continually changing and user responsibilities within the organisation are continually revised. This requires an ability to modify existing role associations at run-time with respect to the current situation and therefore we propose a dynamic role definition model supporting the following:

- acquire new or unassigned roles at run-time (required when a users roles and/or responsibilities in an organisation have changed), using predicates that will be evaluated at run-time.
- dismiss or delegate assigned roles at run-time.

#### 4.2 Dynamic Policy Support

In conventional static policy environments it is very difficult for applications to interwork with new services as they are introduced and for users to respond to events requiring their immediate attention that have occurred outside the scope of their assigned policies. We would like to support a more dynamic policy environment that enables users to more flexibly relate to their assigned policies under certain defensible circumstances. This would involve the following:

- policy containing references to run-time variables or environment.
- policy able to be altered at run-time.
- ability to acquire new or unassigned policy at run-time.
- ability to dismiss assigned policy at run-time (particularly, unfulfilled obligations).

Note that the ability to acquire new or unassigned *policy* at run-time **cannot** be considered to be supported by the provision of an ability to acquire new or unassigned *roles* at run-time. This is because the acquisition of a complete role as opposed to the acquisition of a single policy will often represent an unjustified security hazard.

#### 4.3 Dynamic Conflict Analysis Support

There is clearly little point in supporting dynamic roles and policies without the subsequent support of dynamic conflict analysis of role and policy statements. Dynamic conflict analysis is considered to be a biphasic process of *conflict detection* followed by *conflict resolution*. Specifically, the requirements of dynamic conflict analysis are:

- detection and resolution of incompatible co-existent *role* assignments (which may be required to ensure the user does not operate with a union of privileges obtained from combining a number of roles, determined to be *incompatible*).

- detection and resolution of incompatible co-existent *policies* (which may be required when a user combines roles, which themselves are deemed to be quite compatible, but contain policies which in co-existence are in conflict).

Conflict analysis is a process that needs to occur both statically and as a recurring process at run-time. Static conflict analysis (or compile-time analysis is performed when policies and roles are initially specified) and is useful for detecting conflict that has *actually* occurred, usually due to role and/or policy specification error. Dynamic conflict analysis is useful for detecting conflict that occurs as role membership and applicable policies are altered dynamically, at run-time.

It is useful to make the distinction between static and dynamic conflict analysis as the detection and resolution of conflict can be computationally intensive, time-consuming and hence, expensive and would *preferably* be done statically, off-line. However, not all conflicts can be discovered at compile-time, for example,

**lecturer** *permitted* to update **academic records**. [p1]  
**student** *prohibited* to update **academic records**. [p2]

In this case, unless someone is both a member of the role lecturer and student at compile-time then these policies cannot be considered to be in conflict as they do not currently, and may never at any time in the future, represent an *actual* conflict. It should be noted that it is both possible and indeed desirable, for policies to co-exist in *potential* conflict with each other [p1, p2] without the need for conflict resolution when the policies are defined.

In our dynamic policy model we recognise that as an enterprise evolves, responsibilities alter and consequently, role and policy definitions or assignments may change at run-time, therefore, if a person later becomes both a student and a lecturer, then this would give rise to an actual conflict of [p1, p2] requiring the attention of a run-time conflict detection and resolution process.

## 5 Optimism to Support Trust

Communities within an enterprise often share common organisational goals and should therefore be regarded with a certain degree of trust. It is useful to determine that a community can choose to enforce its policies either by optimistic or pessimistic means [12]. Pessimistic enforcement is inherently preventative (used when trust is low and damage caused by non-compliance is high) whereas optimistic enforcement relies on detection of non-compliance and the subsequent reporting/correction of the non-compliance (used when trust is high and potential damage due to non-compliance is low).

Given that it is not necessarily useful to determine that an *entire* community should be handled *pessimistically* or alternatively *optimistically*, due to differing levels of responsibility, qualification and the nature of tasks within an enterprise; it is reasonable that Policy Authorities should determine for each Policy Space that they manage, whether the entities that belong to it can act independently with respect to the other entities and actions within the policy space (in certain *defendable* circumstances), i.e. *optimistic* enforcement, or indeed, if they should **not** be acting autonomously in any circumstances, hence, *pessimistic* enforcement.

In support of the use of *optimistic* enforcement of policy and roles, we introduce the concept of *Defence*.

The idea of using defences comes from the Common Law in Australia [31]. As citizens of a society we are bound by the laws written down by the parliament of that society. In real-life, however, we sometimes break those laws to achieve a greater good (or in certain acceptable circumstances), this is why Common Law affords its citizens a number of defences in relation to the breaking of set laws [28, 31], some of these are as follows:

- *Necessity* - A person committing a crime because of the necessity to prevent a greater danger may not be punished for such a crime. There are three elements to be satisfied for this defence to succeed, namely (1) the accused must have done the forbidden act to prevent an irreparable evil upon himself or herself or upon others whom the accused was bound to protect; (2) the accused must have been placed in a situation of imminent peril, and (3) the act done must not be out of proportion to the peril to be avoided.
- *Mistake of Fact* - A person who performs an act under an honest and reasonable but *mistaken* belief is not criminally responsible for the act.
- *Compulsion* - A person is not criminally responsible for an act if it is done: (1) in execution of the law; (2) in obedience to the order of a competent authority which that person is bound by law to obey.
- *Extraordinary Emergency* - A person is not criminally responsible for an act done under circumstances of sudden or extraordinary emergency when an ordinary person would reasonably expect to act in the same way. For example, a person who was a passenger in a car who took control of the car when the driver had a heart attack, could later use the defence of *extraordinary emergency* as justification to be acquitted of the crime of drink-driving.
- *Automatism* - Refers to conduct which is automatic or involuntary: “an act which is done by the muscles without any control of the mind”.

- *Provocation* - Refers to conduct where an accused person is provoked into doing the act in question. Three elements to be satisfied for this defence to succeed are (1) the provocation was such that an ordinary person would also have lost their self-control; (2) the accused must have lost control in direct response to the nature of the provocation; (3) the accused must have acted immediately in response to the provocation.
- *Honest Claim of Right* - A person is not criminally responsible for an act if the act is done with an *honest claim of right* without intention to defraud. For example, if you had been entitled to perform some action in the past and had not requested permission to continue performing the action in future (the action having later been deemed to be prohibited), you could plead the defence of *honest claim of right*.
- *Volenti non fit injuria (voluntary consent by plaintiff)* - In cases where the accused can prove that there was “voluntary acceptance of the risk” by the plaintiff, then the defendant will be absolved from all responsibility for the damage caused.
- *Contributory Negligence* - Where any person suffers damage as the result partly of their own fault and partly the fault of any other persons, a claim in respect of damage shall be reduced to such extent as the court deems just, having regard to the claimants share in responsibility for their own damage.

It is stated in Common Law that “a person who is accused of a crime and his or her defence succeeds, that person will be acquitted of the crime” [31]. A defence is a justification or an excuse for an act or omission committed by the accused. Of the defences listed above, we believe that the defences of *necessity, mistake of fact, compulsion, automatism, extraordinary emergency* and *honest claim of right* most closely associate with breaches in enterprise policy. It is currently unclear as to how suitable the remaining defences would prove to be in an enterprise environment.

In our Dynamic Policy-Based Management model, defences may be **specific** or **organisational**. Specific defences are associated uniquely with policies while organisational defences are associated with organisational goals. For example, a specific defence [d1] associated with policy [p7] might be:

**paediatric nurse** *prohibited* to administer non-prescribed medicine **patient**. [p7]

[p7] defence *necessity* **patient in immediate peril**. [d1]

In this case, the **paediatric nurse** may have noticed the child was suffering a from a severe allergic reaction to the

pain-relieving drug she had just administered and in order to save the child's life she would have needed to administer a previously un-prescribed drug to control the reaction and prevent the child's death. In this case she is clearly in breach of her assigned policy [p7] and will need to plead the defence of *necessity*, [d1]. Her breach of policy [p7] could also be supported by the more general organisational defence of *preserving life*.

In order to support trust in an enterprise we have applied notions of *optimistic and pessimistic* determination for policy spaces and introduced the notion of using defences to support breaches to assigned policy.

## 6 Related Work

There exist many architectures that have developed policy-based approaches to the management of open distributed systems. In this section we briefly highlight a selection of the key advances in policy-based management research and compare them to our Dynamic Policy-Based Management model. We discuss Enterprise Modelling in RM-ODP, traditional Role-Based Access Control (RBAC), Policy-Based Management and Collaborative Environments. Our Dynamic Policy-Based Management model allows us to express policies supported by these other models, and it also provides a means to support a more complex, dynamic and open enterprise environment, where policy management must be flexible in order to respond to situational enterprise events.

*Enterprise Modelling: RM-ODP Enterprise Viewpoint.* The group working on the International Standards Organisation (ISO) Open Distributed Programming Reference Model (RM-ODP) [8, 9, 10, 11] are defining an Enterprise language for the purpose of describing constructs such as enterprise objects, object roles and policies which will be used to govern interactions between objects in a community.

Even though the RM-ODP recognises the concepts of roles and policies as a viable means of modelling an ODP system from a high-level enterprise perspective, it has not yet reached a decisive conclusion as to how these modelling constructs would be used to implement a system. That is, the constructs used to describe a system from the enterprise viewpoint are not mapped to the related viewpoints of computational and engineering.

Furthermore, the policy concept is not yet clearly specified, the enterprise model alludes to the ability to express business logic in terms of policies but does not define how the policies will be written, this is the subject of on-going work and discussion [12].

*Role-Based Access Control (RBAC).* Role-Based Access Control [2, 21, 22] brought forward the notions of role and policy to manage database security. RBAC however, pro-

vides limited static policy definition and analysis and limited conflict detection and resolution mechanisms. There is no support for heterogenous distributed domains, compound actions or trust.

*Policy-Based Management.* A substantial body of research on the specification and enforcement of management policies for distributed systems has been performed at Imperial College, University of London [16, 26]. Within their policy-based management architecture, domains have been identified as a means of partitioning objects in distributed systems [29, 30] and management policies [13, 15] were identified as a means of specifying both authorisations and obligations. However, the domain structure represented in this model is more suitable for homogenous, static environments and not to large, evolving enterprises, which we are interested in supporting.

The policy-based management architecture provides a clear policy notation [14], however, policies within this model are necessarily a static concept. Supporting open distributed enterprise relies on defining and analysing policies at run-time, so users are able to interact with new resources for which they have no established policy on access. Furthermore, within this model, conflict analysis is an essentially static concept which would need to be extended to incorporate dynamic conflict analysis in order to support open distributed enterprise.

*Collaborative Environments.* The major contribution of this work [4, 5] is to recognise the limitations of static role definition and to propose a design for the definition of dynamic roles. Roles are presented not only as statically-defined collections of users but also as dynamic descriptions of users that are evaluated as applications are run. Unfortunately, this work does not provide solutions for dynamically dismissing or delegating roles at run-time nor does it consider dynamic policy or the very complex issue of dynamic conflict analysis.

There exist several other policy-based management models but we have selected the above as being representative of the major undertakings and progress in this area of research.

## 7 Conclusion and Further Work

In this paper we have introduced an approach to policy-based management which we believe is more suitable to the management of large, open evolving enterprises.

We have shown that it is possible to support *scalability* within an enterprise through the use of hierarchical enterprise domains, policy spaces, policy authorities and role profiles. System *evolution* is supported through the use of dynamic roles, dynamic policies and dynamic conflict analysis and finally *trust* is supported through defining optimistic policy spaces and the use of defences.

We have introduced a number of novel concepts in support of large, evolving enterprises not considered in current literature. Continued development of these concepts and development of algorithms for dynamic conflict analysis of roles and policies using the enterprise concepts introduced in this paper will be the subject of future work.

## 8 Acknowledgements

The work reported in this paper has been funded in part by the Co-operative Research Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government's CRC Programme (Department of Industry, Science and Resources).

## References

- [1] S. Bettink, K. Connolly, C. Findlay, D. Brennan, American Graduate School of International Management, "Baring Brothers and Company - Technical Note on Rogue Trader Nicholas Leeson", A06-99-0021, USA, 1999.
- [2] F. Chen, R. S. Sandhu, "Constraints for Role-Based Access Control", Proceedings of the First ACM/NIST Role-Based Access Control Workshop (RBAC '95), Gaithersburg, Maryland, USA, December, 1995.
- [3] N. Damianou, N. Dulay, E. C. Lupu, M. S. Sloman, "Managing Security in Object-Based Distributed Systems using Ponder", Proceedings of the Sixth Open European Summer School (EUNICE 2000), Enchede, The Netherlands, September, 2000.
- [4] W. Keith Edwards, "Policies and Roles in Collaborative Applications", Proceedings of the ACM Workshop on Computer Supported Cooperative Work (CSCW '96), Boston, Massachusetts, USA, November, 1996.
- [5] W. Keith Edwards, "Flexible Conflict Detection and Management in Collaborative Applications", Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '97), Banff, Alberta, Canada, October, 1997.
- [6] S. Fay, "The Collapse of Barings", W. W. Norton and Company Publishers, USA, ISBN 0393040550, January, 1997.
- [7] D. Ferraiolo, J. Cugini, R. Kuhn, "Role-Based Access Control : Features and Motivations", Proceedings of the Eleventh IEEE Annual Computer Security Applications Conference, New Orleans, Louisiana, USA, IEEE Computer Society Press, 1994.

- [8] ITU-T Recommendation X.901 — (ISO/IEC 10746-1), International Standard 10746-1, ODP Reference Model Part 1, Overview, June, 1995.
- [9] ITU-T Recommendation X.902 — (ISO/IEC 10746-2), International Standard 10746-2, ODP Reference Model Part 2, Foundations, January, 1995.
- [10] ITU-T Recommendation X.903 — (ISO/IEC 10746-3), International Standard 10746-3, ODP Reference Model Part 3, Architecture, January, 1995.
- [11] ITU-T Recommendation X.904 — (ISO/IEC 10746-4), International Standard 10746-4, ODP Reference Model Part 4, Architectural Semantics, June, 1995.
- [12] P. Linington, Z. Milosevic, K. A. Raymond, "Policies in Communities: Extending the ODP Enterprise Viewpoint", Proceedings of the Second International Enterprise Distributed Object Computing Workshop (EDOC '98), La Jolla, California, USA, November, 1998.
- [13] E. C. Lupu, "A Role-Based Framework for Distributed Systems Management" Ph.D. Thesis, Department of Computing, Imperial College, University of London, UK, July, 1998.
- [14] E. C. Lupu, D. A. Marriott, M. S. Sloman, N. Yialelis, "A Policy-Based Role Framework for Access Control", Proceedings of the First ACM/NIST Workshop on Role-Based Access Control (RBAC '95), Gaithersbury, Maryland, USA, December, 1995.
- [15] E. C. Lupu, M. S. Sloman, "A Policy-Based Role Object Model", Proceedings of the First International Enterprise Distributed Object Computing Workshop (EDOC '97), Gold Coast, Queensland, Australia, October, 1997.
- [16] E. C. Lupu, M. S. Sloman, "Towards a Role-Based Framework for Distributed Systems Management", Journal of Network and Systems Management, Volume 5, Issue 1, Plenum Press Publishing, 1997.
- [17] C. M. Neuwirth, D. S. Kaufer, R. Chandhok, J. Morris, "Issues in the Design of Computer Support for Co-authoring and Commenting", Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW '90), Los Angeles, California, USA, 1990.
- [18] E. Kaethe, (editor), "American Heritage Dictionary of the English Language", Fourth Edition, Houghton Mifflin Company and American Heritage Incorporated, New York, USA, September, 2000.
- [19] Object Management Group, "Unified Modelling Language v1.3", OMG ad/99-06-08, June 1999.
- [20] Oxford University Press, "Oxford English Dictionary", January, 2000.
- [21] R. S. Sandhu, "Role-Based Access Control", Advances in Computing, Volume 46, Academic Press, 1998.
- [22] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, "Role-Based Access Control Models", IEEE Computer, Volume 29, Issue 2, February, 1996.
- [23] C. Schwarz, G. Davidson, A. Seaton, V. Tebbit, W. and R. Chambers Limited and Cambridge University Press, "Chambers Cambridge English Dictionary", Edinburgh, UK, 1998.
- [24] M. S. Sloman, "Distributed Management: What and Why?", Network and Distributed Systems Management, pp. 3-13, January, 1994.
- [25] M. S. Sloman, "Management Issues for Distributed Services", Proceedings of the Second IEEE International Workshop on Services in Distributed and Network Environments (SDNE '95), Whistler, British Columbia, Canada, June 1995.
- [26] M. S. Sloman, E. C. Lupu, "Policy Work at Imperial College: Summary of Policy Work at Imperial College", 1997.
- [27] M. S. Sloman, J. Magee, "An Architecture for Managing Distributed Systems", Proceedings of the Fourth IEEE International Workshop on Future Trends of Distributed Computing Systems, pp. 40-46, IEEE Computer Society Press, September, 1993.
- [28] D. K. Srivastava, T. Deklin, P. Singh, "Australian Law", LBC Information Services, Sydney, Australia, 1996.
- [29] N. Yialelis, "Domain-Based Security for Distributed Object Systems", Ph.D. Thesis, Department of Computing, Imperial college, University of London, UK, August, 1996.
- [30] N. Yialelis, E. C. Lupu, M. S. Sloman, "A Security Framework for Supporting Domain-Based Access Control in Distributed Systems", Proceedings of the IEEE Internet Society Symposium on Network and Distributed System Security, San Diego, California, USA, February, 1996.
- [31] L. Zines, "The Common Law in Australia : Its Nature and Constitutional Significance", Federation Press in Association with the Centre for International and Public Law, Australian National University, Leichhardt, NSW, Australia, 1999.