Governance Policies for Privacy Access Control and their Interactions

Waël Hassan[1], Luigi Logrippo[1,2]
[1]*University of Ottawa, School of Information Technology and Engineering*
[2]*Université du Québec en Outaouais, Department of Computer Science and Engineering*
wael@ieee.org, luigi@uqo.ca

Abstract. We propose the use of process-based access-control methods in the construction of privacy governance systems. Access constraints are specified by policies, but the governance model thus created is prone to interactions and inconsistencies. We show how UML can be used in order to represent the model and how the language Alloy and its model analyzer can be used to formally specify it and to detect interactions and inconsistencies. Examples are taken from the area of banking.

Keywords. Privacy, Business Process, Access Control, Interactions, Alloy, Model analysis, Formal Techniques.

1. Introduction

1.1. Overview and motivation

Seamless governance [24], the ability to govern business and technical requirements of an organization, is the future of enterprise control systems. By using high-level policies, these systems will be self-managing [17]. Most commonly today, high-level policies are associated to enterprise roles. We conjecture that in the future they will be applied to enterprise processes. Governance will be achieved through a homogeneous interface that satisfies business and technical requirements. The interface requires a formal model that verifies inconsistencies and interactions and the respect of operational and business concerns.

Model construction and control of policy systems is a domain dependent problem. These systems are starting to be used in the telecommunications domain, where they are viewed as an evolution of telephony features. Correspondingly, the well-known feature interaction problem is now being generalized to the policy interaction problem [10][22]. It has been noted in some of the above references that the realms of governance and telecommunications policies are converging. Therefore, there is a need for modeling and analysis techniques that can cover both, although a difficulty is the fact that in telecommunications, policies tend to be operational while in enterprise systems, policies are rules of governance.

The convergence of governance and telecommunications policies can be observed in the area of web services. The kind of enterprise model considered in this paper has strong affinities with models found in the business process flow languages that are

being used to define web services, such as BPEL from IBM [30]. The enterprise examples we discuss can easily be adapted to relate to such services. Thus the validation techniques we use have relevance in that area [20].

Privacy control systems, which are a special type of governance systems, currently suffer from the complexity of privacy models, leading to difficulty of verification, since enforcement in privacy is increasingly dependent on business function and human behaviour, where business context (process) has to be considered in issuing access rights. Access rights may depend not only on the role of the person in the organization, but also on the process in which the person is involved at the time of access. Prerequisite for such a policy system is an enterprise control framework that takes into consideration an operational control model. Unfortunately, little has been done to address operational enterprise requirements. [22] suggests a layered architecture for policy environments, however it does not take into consideration business concerns. This paper addresses the issues of formal specification, interaction detection and policy compliance in privacy control systems.

Formal models for privacy policy systems are essential for verification of system properties and detection of interactions [16]. Verifying properties is an important requirement. It is particularly relevant in the privacy domain, as companies need to prove their privacy commitments to their consumers, i.e. a corporation needs to show that its practices are compliant with their published privacy policy. Proving properties guarantees the respect of privacy clauses across the organization. Failure to preserve privacy of consumer data is punishable by law in many countries. Equally important, violation of properties can cause irreparable harm to corporate reputation. The possibility of specifying and verifying systems formally will lead to much tighter and reliable privacy systems than can be considered now. Verifying a system policy can be equivalent to proving the impossibility of some situations. For example, information concerning an address will only be accessed for shipping purposes and not for marketing.

The formulation of enterprise policy systems must be done taking into account the effects that policies can have vertically and horizontally across the organizational structure, leading to the possibility of conflicts or inconsistencies (interactions). Conflicts, moreover, can be a result of policies inherited through business agreements. Clearly, interactions can lead to violation of properties. A designer should be able to verify conflicts, at design time, before policy deployment.

According to [25], reducing complexity requires the reduction of included artifacts and focusing on a single system view. Decoupling of entities and attributes is a common technique. RBAC [12] for example separates between users and access-rights by introducing roles. A role groups users of similar properties in the same logical unit, consequently reducing complexity.

Generally, this work can be broken down into two stages. In the first, we show that using model analyzers such as Alloy we can successfully prove properties of policy systems and detect interactions. In the second stage we will show that our MetaModel is a generic model for privacy systems, and that it can reduce complexity and successfully implement business requirements. In this paper, we will limit the discussion to stage one. Other papers discussing the later stages will follow.

Several privacy languages exist, among others EPAL [4][23] and EPML [4]. Both languages propose a set of templates representing privacy requirements. However, these languages do not have a formal model, therefore system properties cannot be proven, nor can interactions be detected. We believe that the basic ideas of this paper can eventually be applied to other privacy languages, such as these.

1.2. Organization of the paper

The rest of this paper is organized as follows. Section 2, Background and related work explains the concept of process governance, and the basic concepts of the tool Alloy, in addition to presenting related work. Section 3, Process based interaction examples, introduces examples from the problem space, some of which we implement in later sections. Section 4, Formalizing privacy policies, presents a semi-formal representation of the process based governance model using UML, yielding a MetaModel. Section 5, Model abstraction, shows how the MetaModel can be represented in Alloy and presents formal properties of the modeling system and verification tool. Section 6, Results, shows the importance of detecting interactions in the new domain we have introduced, and the capability of using formal methods to address these interactions. Section 7, Discussion, provides insight into process attributes and Alloy issues. Section 8, Summary and future work, resumes the paper and presents future directions.

2. Background and related work

2.1. Organizational governance using processes

High-level feature management requires coordination between operational control and business requirements [18]. Business requirements include workflow in addition to rules and global constraints. [22] refers to high-level requirements, calling them user level interfaces. Business requirements, we suggest, are dependent on an adopted governance model. *Governance* means: the process of decision-making or the process by which decisions are implemented [26]. In other words, governance is the framework of decision-making. In process-oriented organizations, governance dictates that processes are the basic building blocks of enterprises. It also suggests that roles (positions in a company) participate in a process [28]. There are several definitions of what a process is in the business world, most dictate that a process is a sequence of actions [7], and that a process has optional inputs and outputs [14].

We take the view that a business process is a unit that can be composed of steps and/or processes, with the exception of the enterprise process, which is composed solely of processes. Steps in a process are sequenced [13]. The Enterprise process, which is the topmost process in an enterprise, is composed of other processes. In future work we intend to study processes with parallel steps.

For example, *a main process of a banking enterprise includes personal banking, business banking, investment process, and insurance process. An bankbook issuance process includes getting a request, placing an entry, delivering the bankbook. A loan application includes getting a request, verifying credit process, and mailing card.*

1. (Process)Banking:- (Process) PersonalBanking, (Process) BusinessBanking, (Process) Investment, (Process) Insurance.
2. (Process) IssueBankBook:- (Step) GetRequest, (Step) PlaceEntry, (Step) DeliverBook.
3. (Process) LoanApplication:- (Step)GetRequest, (Process) VerifyCredit, (Process) MailCard.

Processes as mentioned earlier represent context [5]. A business context is the environment surrounding a particular activity. A purpose justifies access rights and has a purpose.

> For example, *when we allow Alex to open file F as a part of her organizational role structure, Alex will have access to file F at all times, and can use it regardless of her job function. However, if she was assigned file F as a part of process loan application, then the permission is only available during the sequence of operations leading to a loan application*

Processes are governed by policies such as the above, which can be generic or specific. An Enterprise Wide policy (generic) is one that defines a global requirement and is essentially a system invariant. A generic policy defines global requirements and governs the global process representing the enterprise, therefore its scope is global. For example, *consumer data will not be shared with third party* or *nobody has access to more than one door key*. It may be achieved through sub-policies attached to component processes. A specific policy constrains one such component process. It binds a process to a person, a role, an access-right, or a resource. For example, *role ClientService has access to process IssueCard.* Policies whether generic or specific should be justified by a purpose, and be non-conflicting within their scope.

2.2. Role-Based Access Control

Role-Based Access Control (RBAC) [12][8] is a multi-level security mechanism that associates roles with individual users in order to determine their access rights. In the RBAC framework, users are given roles based on their position in a particular organization. Role-Based Access Control is used in several commercial and government systems. Each organization has its own role ontology. Role-based models can be specific enough to apply to the UNIX model [27], and generic enough to implement business concepts like organizational hierarchy. The essence of role-based access-control lies in the notion of role as an intermediary between subjects and objects: roles are given access-rights to objects while subjects are associated with roles [29]. In other words, the RBAC model separates between users and their privileges by injecting roles in between. The model significantly reduces complexity, resulting in more compact access control policies. These policies are also more generic and portable, since they are not tied to specific individuals.

For any access-control paradigm, administration is an important aspect. A simple model with complex administration procedures is normally unsuccessful. RBAC offers several administrative operations; roles can be added, removed, and delegated. In RBAC, an administrator may subscribe a user to a role. A person can also be revoked from a role. Revocation is the reverse process (un-assigning person from role). Users may delegate their roles to other users. Delegation of roles is a perilous function. It is very practical; however it may cause adverse effects. Delegating a role means that a delegate is assigned role rights and responsibilities.

2.3. Alloy concepts

Alloy [1], consists of a formal language and related model analyzer and therefore defines a formal method, which can be used to precisely capture and analyze logical specifications of systems.

The Alloy language is a simple structural modeling language based on first-order logic. The model analyzer[1] can generate instances of invariants, simulate the execution of operations and check user-specified properties of a model. There are three basic levels of abstraction in Alloy modeling. At the highest level, Alloy follows an object oriented paradigm. The middle level is the set theory level where a model is represented in terms of sets and relationships. At the lowest level are atoms and relations, which correspond to the true semantics of the language.

2.4. Alloy overview

An Alloy model is a set of classes defining the basic relationships. Classes are bound by relationships. Functions and facts set predicates. Finally, assertions are run to verify the validity of a ruleset.

2.4.1. Language constructs

Signatures: A signature is like a class in UML, or record in Pascal. It is a basic unit signifying a relationship with its members.

Relationships: Everything in Alloy is a relationship. Signatures are relationships and can have relationships between themselves. Alloy offers the possibility of discovering reverse relationships using the operator (~). Example: if F: A->B, then ~B is the set of all elements that have a relationship with B in A.

Facts: Facts are uncontested invariants. A system with contradicting facts cannot be instantiated. Facts are asserted as invariants using the keyword fact. Facts can constrain values or relationships.

Assertions: Assertions are used as questions to find out if a rule is violated. This can help detect known interactions. For example, a question can be if it is possible that employees of the credit department have access to the loans department.

2.4.2. Operators

Alloy supports the regular set-theoretical operator set. [+,&,-] set union, intersection, and difference, in addition to the set quantifiers [all, some, none, sole, one]. Most important, we address the dot composition operator [“.”]. The dot operator has multiple significations depending on the level of abstraction used.

Composition [“.”]: The composition operator [“.”] works as a join between two tuple sets. For example, Set $S_1(A,B)$ can be composed with set $S_2(B,C)$ to produce $S_3(A,C)$. It can also be used to dereference a relationship in the class.

```
4.  sig role {
5.      parent : lone role
```

[1] http://alloy.mit.edy/

6. }

where role.parent de-references role. Dereferencing has the same meaning as in the OO model.

Multiplicities: Any relationship can have a multiplicity in Alloy. Multiplicities are crucial because they tell the Alloy analyser how many instances to create. For example, there can be 3 roles, 15 employees and 20 contractors. This instructs Alloy to create such a model with the required constraints.

Scope: Knowing that computing the consistency of the model depends on the arity and the multiplicity of objects, Alloy allows to specify the size of an instance space during execution. This allows users to say: run my world of 5 object classes and 50 instances with their corresponding relationships, to see if the model is consistent.

2.5. Related work

Complex policy systems are prone to inconsistencies that can cause user intentions to be violated. A policy system can contain rules to solve inconsistencies, however it will not be assured that this solution corresponds to user intentions. For example, if a rule takes precedence over another, the second one is violated, which may not be what the user intends. This problem was identified in traditional telecommunication system, under the name of Feature Interaction problem, since these systems offer features to users and these features may interact. With the higher programmability of telecommunications system made possible by internet telephony, and in web services, features will be driven by policies and these may be inconsistent, which means that the feature interaction problem will become a special case of the policy inconsistency problem [22]. Policies are also the basis of business governance. Since access control systems can specify an employee's operational capability, policies can be used to define access rights in a business context.

As mentioned, modeling methods, both formal and semi-formal, can be used in order to detect inconsistencies and other problems in policy systems. However, this requires languages to specify these systems. The work on business processes took a leap forward with the adoption by the standardization group OASIS of the concept of process as a method for capturing business-to-business communication and organizational policy governance. Different XML based languages, including ebXML [11] encourage the use of the process concept. However, the support of formal techniques in defining business processes in order to detect privacy interactions is virtually non-existent, hence the reason for this work.

Most of the work in the formalization of business domains tackles business workflows. There were also several attempts to capture business governance semi-formally. [6] presents an application to constructing systems from process models. It discusses a combination of a UML-based process design language and security modeling language for formalizing access control requirements. Although [6] does not utilize processes as defined in the business domain, its approach may be applicable to workflow systems. Another issue with this paper is that it is not applicable to privacy aware systems, but it is aimed particularly at security protection environments. In [20], the authors suggest a requirement driven approach to the design and verification of web services. They also suggest the use of model checkers to verify system constraints. This approach comes closer to the business domain, which makes it suitable for web

services environments, but does not address interactions, and is not adapted to privacy systems. [15] uses the concepts of purpose and attempts to create a purpose hierarchy, as well as a model that enforces privacy.

3. Process based interaction examples

What are governance policy interactions? How can they occur? And how can we detect these interactions? How can business processes and the process based model help reduce these interactions? Separation of concerns and delegation of authority are two models of interactions that exist in several domains.

3.1. Process hierarchy

For the two next subsections consider the following process hierarchy.

7. (Process) CreditCardApp:- (Process) ReceiveCardApplication, (Process) CallCreditCheck, (Process) IssueCard, (Process) CreateAccount.
8. (Process) CreateAccount:- (Step)LeaveTraceInSystem, (Process) CreateCard, (Process) MailCard.
9. (Process) DeleteAccount:- (Step)LeaveTraceInSystem, (Step)RemoveAccount.
10. (Process) WithdrawApplication:- (Process) DeleteAccount, (Step) NotifyClient.

Process Credit Card Application includes four processes (line 7). Process CreateAccount also includes one step and two processes (line 8). Process DeleteAccount includes a step called LeaveTraceInSystem and another step RemoveAccount. Each one of these processes can be assigned to different employees, according to company policies.

3.2. Separation of concerns

The concept of separation of concerns is important in the banking industry. One of its applications is that a principal is not allowed to combine access to specific functionalities. Should these capabilities be combined, a violation of a policy may occur. For example,

Privacy Interaction Example:

Policy P1: An employee cannot have access to both customers' address and credit card information(Card Number, expiry date, PIN, and last 4 digits on the back of card) ;

Such a rule is general and applies to the credit card application process, see line 7. However, one of the tasks of issuing a new card (CreateAccount), see line 8, includes the mailing of the credit card to the consumer. Given this requirement, assigning to an employee the process of line 7, as well as to the process of line 5, will violate the conditions set by P1. In this scenario, a local policy violates a more general policy

Security Interaction Example:

P_2: An employee cannot have access to both CreateAccount and DeleteAccount processes. This prevents people from misusing the system to find information about a user.

When an employee has access to both processes in WithdrawApplication, line 10, and CreateAccount, line 8, rule P2 is violated.

3.3. Delegation example

The following example describes a situation where a delegation of tasks causes a conflict of permissions:

Privacy Example: *In a company, the LoanProcessing process includes the VerifyCredit process. However information collected for the purpose of credit verification should not be available to employees doing loan processing. Suppose now that an employee assigned to sub-process Verify-Credit goes on vacation and delegates his rights to his manager, who is a member of the process Loan-Processing. The manager receives access to information that should be denied to her.*

This interaction is caused by the fact that data collected for the purpose of credit verification can be used for another purpose.

4. Formalizing privacy policies

Privacy specification, like security specification, can be included in system design at a high level of abstraction. We present a formal privacy model for a process-governed enterprise. We provide a UML MetaModel together with a translation into an Alloy model. Using our MetaModel, we make it possible to develop privacy aware systems that are designed with the goal of preventing violations of privacy policy.

We conjecture that an organization can be formally represented as a composition of its constructs, which become its distinguishing attributes. Our MetaModel acts as a stereotype from which instance models are inherited. This MetaModel constrains instance models to contain basic enterprise constructs: roles, persons, devices, processes and activities, and policies. Among related work presenting a similar list, [15] excludes processes; however it includes domain, subject, object-type, and object classes.

Every access-control model has its basic atoms. For example, in the UNIX model the atoms are the name of the user, an object and access-right. In the RBAC model they are the user-group, object and access-right. Since most organizations are built as a function of roles, it is important to include role as an attribute in any organizational structure. As enterprises become increasingly modeled from a user (client) perspective, business processes compete with roles as basic units of organizational structure. Therefore, in this work roles will coexist with processes.

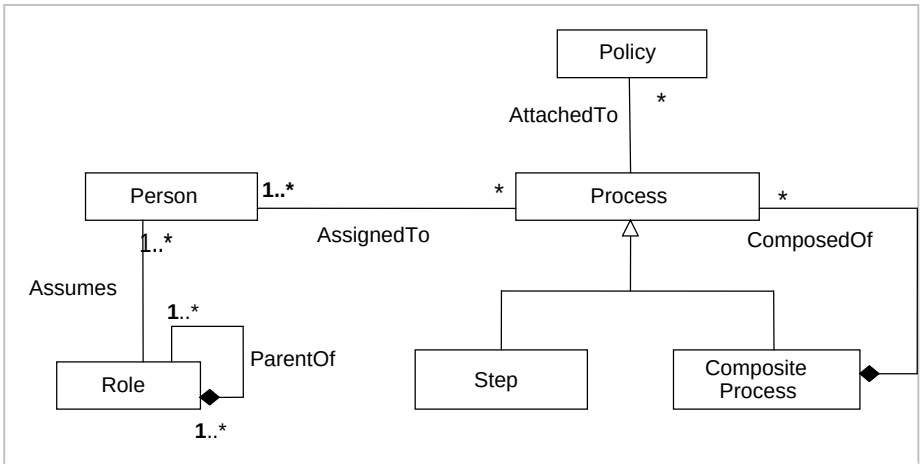

Figure 1 Process Based Enterprise Model

Roles are important because they group properties of persons participating in one role. Roles are functional in nature, and have been used extensively in structuring organizations. Traditionally policies apply to roles. We conjecture that the concept of process offers complementary advantages to the concept of roles, and therefore including processes in enterprise models is a necessity.

Our MetaModel represents the requirements: its main class types are policy, process, step or activity, person, and role. A MetaModel defines the syntax of a class of models, it does not refer to a particular platform. A platform is an execution or verification environment (Alloy in our case). Depending on the enterprise context, a specific model can inherit from a MetaModel. For example in one system, access-rights are bound to processes; in another system, they can be bound to steps. Several implementations can be derived for a given model. For example, one can create two implementations of a financial institution, a bank or a broker.

4.1. MetaModel and Alloy implementation

Models in general are abstractions of the real world and are used to precisely and often formally describe and analyze the working of some relevant portions of the physical system. The main benefit of using models is that they help abstract away irrelevant details while highlighting the relevant [2]. A MetaModel acts as a supertype from which an instance model can be created. A model instance should follow model constraints.

We present a MetaModel for enterprise architecture. The model is generic in its structure and is applied to a banking system. In this section, see Figure 1, a MetaModel includes the following stereotypes: role, person, process, steps, and policy. It acts as supertype from which an instance model can be created. A model instance should define model constraints.

4.1.1. Person

It represents a human resource, an employee, a consultant, and possibly collaborators.

```
11.  abstract sig person{
12.  assumes    : role
13.  assignedTo: Process
14.  }
```

Example: Contractor, Employee, Supplier, etc...

4.1.2. Process

As discussed above, processes involve a collection of people, and policies needed to perform the function. A process, as we defined in section 2.1, can be composed of other processes, or steps. The composition operator signifies a hierarchy of processes. For example, the personal banking service offers manual transactions. These services are broken down into activities, such as deposit, withdraw, verify, provide information, etc. Note that process composition does not suggest any ordering.

```
15.  abstract sig process {
16.      parent         : lone process,    // pointer to a parent process
17.     composedOf:  set steps          // a process is composed of some steps
18.  }
```

Notice that in line 17, steps are defined to be parts of sets. Future work should go into studying sequences of elements in a particular flow.

4.1.3. Steps

Steps are granular elements of a process. These steps are performed by persons, see Figure 1. The “Assigned To” relationship embodies the connection from processes to persons and hence from steps to persons. In practice, this is how enterprises work: management assigns human and physical resources to specific functions. Then it assigns particular tasks to people. Note that a step is defined as an empty unit for simplicity in order to allow Alloy to randomize it.

```
19.  sig step{
20.   }
```

4.1.4. Policy

Policies are usually attached to one process. We can specify a particular instance of a policy and assign it to a process.

```
21.  abstract sig policy {
22.      attachedTo  : lone  process,     //  a policy is attached to one process
23.      permitted: role  -> process,      // permits role access to a process
24.      denied    : role  -> process       //denies a role to a process
```

```
25.  }{
26.       no permitted & denied          //no dual permission and denial to the same object
27.       role.permitted in attachedTo   // Permitted roles have to belong to the attached
     process
28.       role.denied in attachedTo    // Denied roles have to belong to the attached process
29.  }
```

Each policy is attached to one process, see line 22. Lines 23,24 propose a set based method of representing access control. This can be read as follows: permitted is a relationship between a role and a process; therefore, only where a policy is attached to a process can a permission or denial rule apply. Moreover, in no situation, line 26, should there be a permission and a denial for a pair (role, process)

4.1.5. Roles

In our work we suggest that Roles are a total order hierarchy, see line 30. They have a "parent" relationship, in line 31. A role can be a superset of roles and is usually managed by other roles, with exception of the topmost role. There are situations where one role is "child of" multiple roles, from different departments.

```
30.  sig role {
31.     parent : lone role
32.  }
```

Our instance model suggests that an access right is a relationship between a role and a process, as it has been seen in lines 23,24. A variation of the model could assign access rights to steps, something that we may explore in future work, see the AccessTo relationship in Figure 1. In this model however, we choose to implement the role to process mapping and not the role to step. Normally (in RBAC for example) access rights are associations from roles to resources [8].

Example: *Manager, Employee, Director, President.*

5. Model abstraction

We show the implementation of the model in Alloy. The model is in essence an abstraction of processes, policies, relationships, and their respective hierarchies. It is a refinement of the UML representation. In privacy governance, there are a few principles that are the basis of any privacy model. These principles fall under access-control. e.g. separation of concerns, delegation, and definition of inconsistencies. In the next section, we show how to model delegation and separation of concerns and provide examples.

5.1. Representation of access control

The method of abstraction of access-control is a distinguishing feature in any access-control model. The data type and even the implementation are important to its

execution. Access-rights are represented as relationships. This optimizes the clarity and ease of detecting interactions. Access-control atoms can be represented in a logical tuple-space with a set-based interface e.g. (Luigi, Permitted, Door). Using our model, we utilize the power of set relationships and set theory. Set operations provide us the operations of intersection, difference, etc. Each of these is equivalent to a query in a database system. For example, see lines 33-37.

```
33.  abstract sig policy {
34.      attachedTo  : lone  process,
35.      permitted: role   -> process,
36.      denied    : role   -> process
37.  }
```

As mentioned in the UML MetaModel, a policy attaches to a process. Moreover, roles can access processes or activities. In English the above lines read:

A policy is attached to one process. Each policy includes an AttachedTo relationship that connects roles with processes. Access rights (Permissions or Denials) are relationships between roles and processes under a policy.

In this way we represent the main concepts of process-based access-control. This decouples roles from objects and hence facilitates management in a business process context.

5.2. Detection of direct conflicts

Direct conflicts of access-control are restricted by preventing any person from having an acceptance or denial for a single process. Line 43 says that for no policy; the same role will have conflicting access rights.

```
38.  abstract sig policy {
39.      attachedTo  : lone  process,    //  a process is attached to policies
40.      permitted: role   -> process,     // permits role  access to a process
41.      denied    : role   -> process      //denies a role  to a process
42.  }{
43.           no permitted & denied
44.  }
```

5.3. Representation of separation of concerns

We present a solution for the separation of concerns problem by reversing relationships of permission and checking their interaction with the conflicting function. This solution is specific but we plan to extend the model to support separation inherently.

```
45.  assert seperateConcerns {
46.   no (process.~(accountDeleteAP.permitted)  & process.~(accountCreateAP.permitted))
47.  }
```

In line 46, we represent the separation of concerns between Account-Create process and Account-Delete process, We say that processes which receive a permission from an AccountDeleteAP should not be permitted to AccountCreateAP. The ~ operator is a reverse function from process to accountDeleteAP.

5.4. Representation of delegation

As in the case of the separation of concerns issue, we represent delegation by assigning people to processes.

5.5. Proving compliance

A privacy system P is compliant with policy set S when every policy in S holds for P. We use assertions to check if the model is compliant. An example is in lines 45-47 above.

6. Results

In modeling, conflicts are easily created, however they are hard to detect, where by conflict we mean an inconsistent model or a set of constraints that cannot be simultaneously satisfied. Each modeling method requires its own tool for conflict detection.

Conflicts exist in policy systems because of several reasons. The first is that policies do not change the basic principle of atomic access control. The second is that composition of possibly interacting system creates an interacting system. Therefore, existing interactions of low-level atomic transactions translate into policy interactions.

6.1. Example involving separation of concerns

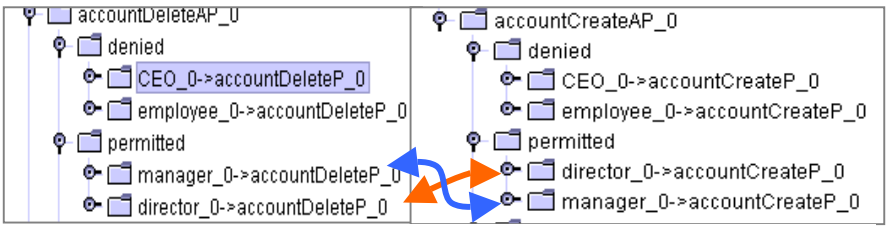

Figure 1. Counterexample

Using Alloy we were able to represent the example presented in 3.2(b), detect the interactions and find counterexamples.

The representation of separation of concerns of section 5.3 reads: no one should be able to create accounts and delete accounts at the same time. Alloy was able to find a counterexample, proving that this indeed is possible in the model.

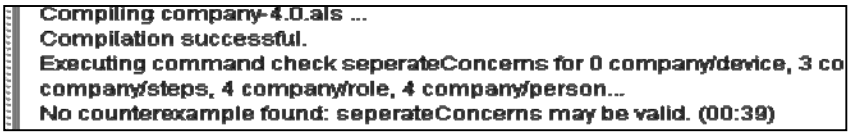

Figure 2. No Counterexample Found

In the counterexample, shown in Figure 1, a manager is permitted to delete an account and to create an account. The director can do the opposite, i.e. delete Account, and Create account.
When we added a separation of concerns policy, the violation ceased to exist, Figure 2. The figure shows the result after Alloy was instructed to separate concerns between two processes (account Create and Account delete). In the allotted scope, the analyzer was not able to find any counterexample.

6.2. Delegation

We dealt with the delegation problem in a similar fashion, and the Alloy analyzer succeeded in providing a counterexample. We do not discuss in this paper the interaction given in section 3.3.

7. Discussion

7.1. Process approach

Interactions occur independently of the choice of hierarchy type, whether of roles or of processes. However, process structure simplifies the specification of privacy systems by adding context. The distinction between process-based and role-based access control is related to flow since role hierarchies do not use flow to assign access rights. Processes, on the other hand, do so by sequencing actions. For example, *a role x can be allowed action B only after having performed action A.* In RBAC, role structure defines the outcome of access policy. In our model, access-rights can be derived from the fact of belonging to a process. Nevertheless, the issues of separation of concerns and delegation are present in both paradigms.

7.2. Alloy characteristics and Issues

The rule-based nature and object orientation of Alloy are suitable for the enterprise model under consideration. A very important property of Alloy is the fact that it allows to limit the instance size. On the other hand, we emphasize the fact that our process based methodology and our claims are tool independent, and can be investigated using other analysers.

7.2.1. Explosion avoidance

A special feature of Alloy is that it can populate a world of possibilities that is constrained by a specific size. One can then model a system with a fixed number

of entities. A system designer can leave it up to Alloy to populate instances randomly and nondeterministically, see lines 48-49.

48. run example for exactly 3 process, 3 policy, 6 steps, 4 role, 4 person
49. run example for 6
50. check separateConcerns for exactly 3 process , 3 policy, 6 steps, 4 role, 4 person

However, an inconsistency that does not show in a certain model size could very well appear in the next one, hence it is important to try as large a size as possible.

7.2.2. Model consistency

Given a model, in Phase 1, if it is inconsistent, Alloy can detect the inconsistency but it does not specify the reasons for inconsistency, nor does it provide a counterexample. Therefore, when building a model, one needs to incrementally verify the system, i.e. the test for consistency must be repeated every time the model is changed, otherwise the reason for the inconsistency can become difficult to detect. It is left to manual labor to decide the cause of the inconsistency.

7.2.3. Model compliance

In phase 2 (corresponding to Line 50), Alloy can check if the policies comply with the model. In the affirmative, it asserts their validity, otherwise it produces a counterexample. In the case of privacy systems, an enterprise needs to commit to a set of privacy rules for its patrons. Such rules can be translated into Alloy and submitted for verification, and Alloy can determine whether these rules are consistent with the model.

8. Summary and future work

We adopted the concept of process as a basis for structural composition of an enterprise. Governance of such process-based enterprises requires a methodology for modelling and a method of verification. We have given a UML MetaModel to specify such enterprises. We have shown our ability to translate the MetaModel into Alloy for verification and interaction detection. Examples in the areas of banking were used, and the extension to consider web services seems to be straightforward. The MetaModel helps translate enterprise structure into Alloy, which has object-oriented semantics. We have then shown by example the ability to model the separation of concerns and the delegation principles. Note that the three examples given in Section 3 were analyzed using Alloy, which was able to detect the conflicts. Eventually, such methods could be used in enterprises to validate their governance models, including their policies.

Future work will attempt to completely formalize the method, combining the capabilities of RBAC as well as other access control models. Larger systems with more constraints will also be studied. We will also address dynamic interactions, which will depend on a state model. The consideration of complex enterprise models will require the specification of ontologies, to describe relationships between concepts and entities in the models.

We plan to create a privacy policy language that can be automatically translated into Alloy for immediate validation. It will be possible to translate this language into established policy languages such XACML.

9. Acknowledgment

The authors are grateful to the Natural Sciences and Engineering Research Council of Canada for partial funding of this research.

References

[1] Alloy Publications. http://alloy.mit.edu/beta/publications.php. Accessed Dec. 2004.
[2] A. Agrawal. Model Based Software Engineering. Graph Grammars and Graph Transformations. Area Paper. Vanderbilt University. EECS. April 8, 2004. http://www.isis.vanderbilt.edu/view.asp?CAT=1&GID=171, Accessed March 2005.
[3] D. Amyot, T. Gray, R. Liscano, L. Logrippo. Interactive Conflict Detection and Resolution for Personalized Features. Submitted for publication.
[4] Automating Privacy in the Enterprise. http://www.zeroknowledge.com/business/. Accessed Jan. 2004.
[5] P. Balabko, A. Wegmann. Context Based Reasoning in Business Process Models. pp. 128. 2003.
[6] D. Basin, J. Doser, T. Lodderstedt. Model Driven Security for Process-Oriented Systems. SACMAT. 100-109. 2003.
[7] A. Charfi, M. Mezini. Hybrid web service composition: business processes meet business rules.. International Conference On Service Oriented Computing. 30- 38. 2004.
[8] S. Coughlan. Role-based Access Control- A Users Guide.. https://www.eema.org/idsol/coughlan.pdf. Feb 2003.
[9] V. Devedzic. Understanding Ontological Engineering. Comm. ACM 45(4). 136-244 April 2002.
[10] P. Dini, A. Clemm, T. Gray, F.J. Lin, L. Logrippo, S. Reiff-Marganiec. Policy-enabled Mechanisms for Feature Interactions: Reality, Expectations, Challenges. Computer Networks. 585 – 603, 2004.
[11] EbXML. Electronic Business using eXtensible Markup Language. http://www.oasis-open.org/home/index.php. Accessed Aug. 2004.
[12] D. Ferraiolo, D. Kuhn, R. Chandramouli. Role-Based Access Control. Artech House. 2003.
[13] M. Gruninger, C. Schlenoff , A. Knutilla, S. Ray. Using process requirements as the basis for the creation and evaluation of process ontologies for enterprise modeling. ACM SigGroup Bulletin, Special issue: Enterprise modeling: notations and frameworks, ontologies and logics, tools and techniques. 52 – 55. 1997.
[14] M. Hammer. The Agenda: What every business must do to dominate the decade. Random House. New York. 2003.
[15] Q. He. Privacy Enforcement with an Extended Role-Based Access Control Model. North Carolina State University Computer Science Technical Report TR-2003-09. 2003.
[16] G. Karjoth and M. Schunter. A Privacy Policy Model for Enterprises. 5th IEEE Computer Security Foundations Workshop. 271-281. 2002.
[17] J. Kephart, D. Chess. The vision of Autonomous Computing. Computer Journal, 41-50. Jan. 2003.
[18] B. Medjahed, B. Benatallah, A. Bouguettaya, A. H. H. Ngu, A. K. Elmagarmid. Business-to-business interactions: issues and enabling technologies. VLDB Journal. The International Journal on Very Large Data Bases archive. 59 – 85. May 2003.
[19] National Computer Security Center (NCSC). A guide to understanding discretionary access control in trusted system. Report NSCD-TG-003 Version 1. Sep. 1987.
[20] M. Pistore, M. Roveri, P. Busetta. Requirements-Driven Verification of Web services. Electronic Notes in Theoretical Computer Science. To appear.
[21] R. Paul, G. Giaglis, and V. Hlupic. Simulation of Business Processes. American Behavioral Scientist. 1551-1576, 1999.

[22] S. Reiff-Marganiec and K. Turner. A Policy Architecture for Enhancing and Controlling Features. Proc. Feature Interactions in Telecommunication Networks VII. 239-246. IOS Press. Amsterdam. June 2003.
[23] M. Schunter, ed.. Enterprise Privacy Authorization Language (EPAL 1.1), http://www.zurich.ibm.com/security/enterprise-privacy/epal, Accessed Jan. 2004.
[24] S. Sunrano. Globalization and Information Technology: Forging New Partnerships in Public Administration. Asian Review of Public Administration, XIII. 2 July-December 2001.
[25] V. Thurner. A formally founded description technique for business processes. Technical Report, Technical University of Munich. Germany. 1997.
[26] United Nations, ESCAP. What is good governance., http://www.unescap.org/huset/gg/governance.htm, Accessed Aug. 2004.
[27] UNIX protection model, http://cs.gmu.edu/~menasce/osbook/. Accessed Aug 2004.
[28] B. Warboys. Reflections on the Relationship Between BPR and Software Process Modelling. LNCS, Vol. 881. Proceedings of the13th International Conference on the Entity-Relationship Approach, Springer-Verlag, UK. 1994.
[29] J. Zao, Hochtech Wee, J. Chu, D. Jackson. RBAC Schema Verification Using Lightweight Formal Model and Constraint Analysis. http://alloy.mit.edu/contributions/RBAC.pdf, Accessed Oct 2004.
[30] Business Process, Execution Language, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/, Accessed 2005.