CERIAS Tech Report 2007-08

A FRAMEWORK FOR SPECIFICATION AND VERIFICATION OF GENERALIZED SPATIO-TEMPORAL ROLE BASED ACCESS CONTROL MODEL

by Arjmand Samuel, Arif Ghafoor, Elisa Bertino

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

A Framework for Specification and Verification of Generalized Spatio-Temporal Role Based Access Control Model

Arjmand Samuel*, *Student Member, IEEE,* Arif Gahfoor†, *Fellow, IEEE,* and Elisa Bertino‡, *Fellow, IEEE*

Abstract

We are witnessing an exponential growth in the use of mobile computing devices such as laptops, PDAs and mobile phones, accessing critical data while on the move. The need to safeguard against unauthorized access to data in a mobile world is a pressing requirement. Access to critical data depends on users' identity as well as environmental parameters such as time and location. While temporal based access control models are well suited for enforcing access control decisions on fixed users, they loose their effectiveness when users employing mobile computing devices are not fixed in space and are moving from a secure locale to an insecure one, or vice versa. Issues of location as a context parameter for access control have been addressed by a number of researchers but definition of rich spatial constraints which effectively capture semantics and relationship of physical and virtual (e.g. membership to an IP group) locales is still missing. The inclusion of multiple constraints (temporal and spatial) to the access control policy exposes the need to be able to compose a policy which is verifiable for consistency and structural integrity. Further, the access control policy is expected to evolve over time and inclusion of new constraints, permissions or user rights may conflict with the existing ones. In this regard, we draw upon techniques developed for software engineering and use them for policy specification modeling and conflict resolution. The first contribution in this paper is the development of the Generalized Spatio-Temporal Role Based Access Control (GSTRBAC) model, by proposing a formal framework for composition of complex spatial constraints exploiting topological relationship

*Purdue University, Electrical and Computer Engineering; West Lafayette, IN 47907. amsamuel@purdue.edu. Corresponding Author

†Purdue University, Electrical and Computer Engineering; West Lafayette, IN 47907. ghafoor@purdue.edu.

‡Purdue University, Department of Computer Science; West Lafayette, IN 47907. bertino@purdue.edu.

between physical and virtual locales. Spatial constraints are defined for spatial role enabling, spatial user-role assignment, spatial role-permission assignment and spatial activation of roles. The notion of spatial separation of duty is also developed whereby a user is not permitted to activate two roles simultaneously if the roles are being activated from specific locales. Another feature of the proposed GSTRBAC is the spatial role hierarchy, which allows inheritance of permissions between roles, contingent upon roles being activated from predefined locales. The second contribution in this paper is GSTRBAC policy specification framework using light weight formal modeling language, Alloy and, analysis of access control policy model using the accompanying constraint analyzer. In addition, for consistent evolution of access control policy, the policy administrator can specify additional policy fragments in the policy model and can verify consistency of the overall policy for conflict free composition of the actual policy.

I. INTRODUCTION

The need to safeguard against unauthorized access to data by users using mobile devices such as laptops, PDAs and mobile phones bring to the forefront the requirement for incorporating spatial context to access control mechanisms. While using temporal based access control mechanisms was considered sufficient for fixed users, it has proven to be ineffective for mobile users because security characteristics of one location are different from another and uniform access to secure and privileged information every where is not a realistic solution.

A motivating example in this regard is that of a physician accessing Electronic Health Records (EHR) of a patient. Complete access to designated portions of EHR is granted to him while he is at his clinic. However, the same physician accessing the same part of EHR from his home may not be allowed the same level of privileges since the privacy of the patient may not be protected from his home, as it is from the hospital. Similarly, a consultant may have full access to records relevant to the company he is consulting to, as long as he remains within the confines of the company offices. For reasons of privacy and business competition, he may not have any access to the same records when he is at home or at the premises of another client.

Location as a context parameter exhibits a number of characteristics making its use in access control decisions challenging. Prominent among them is heterogeneity; the fact that location can be referred to as a noun (name of a city) or as a collection of numbers (longitude, latitude). Another challenge is the fact that multiple locations are semantically related to each other (city is in a state, which in turn is in country) and arriving at a standard relationship may not always

be that obvious.

The need to develop a fine-grained access control mechanism which utilizes spatial information has been pointed out by a number of researchers, such as [1]. Role Based Access Control (RBAC) has emerged as a de-facto model for specifying security requirements of large organizations. Its strength lies in the definition of user roles more akin to the functional responsibilities of users in the organization and abstracting object permissions as roles [2]. The Generalized Temporal RBAC (GTRBAC) incorporates a set of language constructs for the specification of temporal contextual constraints [3]. However GTRBAC does not allow the specification of rich spatial constraints in which relationship between spatial contexts effect access control decisions. In the current paper we extend GTRBAC model to include rich spatial constraints and define spatial role hierarchy and separation of duty. We call this extension the Generalized Spatio-Temporal Access Control (GST-RBAC) model.

Typically a set of access control directives is referred to as a policy. Policies are defined as *information which can be used to modify the behavior of a system* [4]. Recently, the use of policies in the context of network and distributed applications and services has been an area of active research and is considered as an emerging software domain [5]–[9]. Policies have been applied to access control mechanisms [10], to Policy Based Network Management Systems (PBNM) [6], [11], to express the enterprise viewpoint [12] of ODP (Open Distributed Processing), to agent based systems [13], and to Web Services Policy (WS-Policy) [14]. Our contribution regarding spatial extension of GTRBAC can clearly spill over to the use of spatial contextual parameters in the above mentioned areas as well.

In order to effectively compose consistent and verifiable policies, we borrow techniques from software engineering and apply them for access control policy composition and conflict resolution. The nexus of software and policy specification results in a generic methodology which is rooted in a known discipline and is aimed at solving an emerging challenge. Specifically, we propose a framework for specification of GST-RBAC access control policies using the lightweight formalism provided by Alloy [15]. Alloy has been effectively used to model structural behavior of a number of software systems [16]. Our proposed specification model formally captures the policy constraints and assertions and provides a conflict resolution mechanism using the Alloy constraint analyzer, which can effectively be used for conflict resolution. It may be noted here that removing conflicts from a policy specification results in an artifact which is consistent and

has been verified to exhibit conflict free behavior, a process known to the software engineering community as verification.

The remainder of this paper is organized as follows. Section II provides motivation for spatial extension of GTRBAC and for specification modeling of GSTRBAC policies. In this section we also discuss the challenges of using location as a context parameter for access control and set the stage for the rest of the paper by enumerating goals for both spatial access control and policy specification modeling. Section III provides the background for the rest of the paper. Section IV defines the concepts to be used later in the paper. We propose the generalized spatio-temporal role based access control model in Section V and discuss its conflicts in Section VI. Our discussion of GSTRBAC is finally exemplified with an example in Section VII. We introduce the Alloy modeling language in Section VIII along with a step by step methodology for the creation of a specification model for GSTRBAC, complete with its conflict resolution and verification methodology. Section IX provides insight into similar work done by other researchers for the sake of providing background information and pointing out research challenges that we have addressed. Finally, Section X concludes this paper by outlining some limitations of our approach and venues of future research.

II. Motivation and Goals

As outlined in Introduction, we now provide detailed motivation for a spatially and temporally aware access control model and define goals of such an access control model. Next, we motivate for development of a access control policy specification and conflict resolution framework.

A. Challenges of Spatial Context Parameters

The importance of context parameters in computing has been emphasized by a number of researchers such as [17]. The use of time as one of the context parameters involved in access control decisions has been a topic of research for some time. While authorization for use of resources based on time allows organizations to exercise fine grained temporal control, the importance of spatial context is also being realized lately. This realization has been amplified with the advent of mobile computing devices utilizing wireless networks and allowing users to access computing resources while on the move.

Increasingly, spatial context parameters are being used to tailor content for mobile users. A user traveling through a city may be presented a list of surrounding restaurants and gas stations, thus allowing selection of the same based on the user's location. It is also possible to provide spatially tailored content to stationary users who request resources from disparate locations. A user may have access to his companies accounts when he logs into the system from his office, but on the other hand, he may be denied access to all of the information when he logs into the system from his home. In this case the location may be sensed by employing physical location sensing technology, or may be sensed using virtual location such as membership to a specific IP address pool.

Unlike time, location parameter can be defined using a number of representations such as geometric, symbolic and virtual. In addition, locations may be related to each other in more than one way, such as a university may be contained in a city and symbolically referring to the university implies being in the city as well. The key challenges posed by location context parameter are discussed below.

Heterogeneity: Location as a context parameter can be represented in a number of formats. Prominent among these representations is the symbolic and geometric. Symbolic location can simply be defined as the names given to locations such as cities, roads etc. On the other hand, geometric location is the location attribute defined in a coordinate system, in two or three-dimensions. There are a number of widely recognized coordinate systems such as cartesian coordinates, polar coordinates, earth centered north pointing coordinates, to name a few. With location being represented in such a wide variety of formats, interpreting and using location context by applications posses a heterogeneity challenge.

Scalability: While it is desirable to be able to apply location constraints to a computing environment with the highest possible resolution, issues of scalability do not allow this. Scalability of location context is further exacerbated when using one form of location representation rather than the other. Various forms of location representation are discussed in Section IV.

Non-monotonicity: Unlike time, which is increasing monotonically, same values of location context can be revisited by the computing application. Non-monotonicity of location context makes its use in the computing environment more complex.

Flexibility: While a number of researchers have proposed the use of location context in the computing environment, a globally accepted representation remains a challenge [18]. A flexible

and globally usable schema for location context will allow different applications to use location context without the need to translate it to an acceptable format.

B. Spatial Extension of Temporal Access Control Model

Applying spatial constraints for access control requires incorporation of spatial parameters in a formal access control model. In order to achieve this we extend the GTRBAC model and introduce a mechanism for defining spatial parameters in constraints. Our rationale for extending GTRBAC for spatial constraints is outlined next.

While temporal constraints for resource authorization may be sufficient for most fixed application in which users assume similar access rights no matter where they are located, spatial constraints are required in applications with mobile users or users with access control rights varying with location. In addition, a flat location constraint may suffice to represent simple locations with predefined resolution but a multi-resolution and hierarchical definition of spatial constraints is required to capture all possible spatial scenarios that may arise when the application is deployed.

In practical application environments, location and time both play a vital role to effectively manage enterprise resources. For this reason instead of proposing a spatial access control model, we propose an extension to GTRBAC, which already provides a well-defined temporal framework.

C. Goals of a Spatio-temporal Access Control Model

Extension of GTRBAC to include spatial constraints is based on a set of goals which are outlined below. As we discuss in the related works section (Section IX), to the best of our knowledge, no single access control model satisfies all of those goals and consequently poses an opportunity for a comprehensive spatio-temporal access control model. A spatio-temporal access control model must be able to

- define time and location based constraints;
- exploit the semantic relationship between locations to compose rich spatial constraints;
- scale up with the increase in number of locations, implying using inter-location semantics to control the number roles in the system.

While the above remain the stated goals of this paper, a number of related issues also surface which are detailed next. The issue of location sensing with appropriate resolution presents a research challenge which has been addressed by a number of researchers [19]–[21] and will be outlined in the related works section (Section IX). This paper builds on top of the techniques developed in the mentioned references. The issue of relationship between locations has been researched in the Geomatics literature and we build on work reported in [22].

D. Goals of Policy Specification modeling and Conflict Resolution of Access Control Policies

The composition of access control policies is an activity performed by policy administrators of an organization. Since this is inherently a labor intensive activity usually composed and implemented in piece meal, chances for introduction of conflicts are very high. Conflicts introduced in a large and practical policy may not be obvious to human inspection but at the same time may cause serious violations of organizational security doctrine. The problem is further exacerbated by the fact that access control policies of organizations evolve over time and addition/deletion of policy fragments may introduce conflicts.

Creating an access control policy specification model allows design of a consistent policy model and helps to uncover design flaws. Conflict resolution is also possible if the specification modeling environment exhibits some form of formalism. Key research challenges in specification modeling and conflict resolution of access control policies arise from the need to,

- model access control policy specifications for verification of policy design;
- model an instance of the access control policy incorporating time and location;
- provide the ability to support light weight formalism for effortless composition, visualization and conflict resolution.

III. Preliminaries and Background

Location is traditionally defined as a place or site in physical space expressed relative to the position of another point or thing. Recently the applicability of location context in computing is highlighted with the advent of mobile and ubiquitous computing, where a user may be provided varying levels of service depending on location. In order to set the stage for using spatial contextual parameters for access control, we provide an overview of spatial representation defined by the community [1], [23].

A. Spatial Characterization

1) Physical: Physical location is the most widely used representation for location. It can include names of places (Miami, New York etc) or can include names of places by their functions (stadium, mall etc). Additionally, physical location can be represented by using various types of coordinate systems in terms of numbers and elevation etc. In this context physical location can be represented according to three dimensions, namely longitude, latitude and height. Physical location can be divided into symbolic and geometric representations. When a physical location is represented by referring to its name or some property, it is said to be symbolic location. A more rigorous definition is stated in Section IV. Symbolic location can be represented by using nouns (e.g. name of a city) and hence is easier to implement. However with a large number of symbolic locations it offers a scalability challenge in terms of the number of names that can be given to physical locations. Geometric location, which is representation of physical location using some coordinate system, is by far much more accurate and offers higher resolution. It is also universal if the coordinate system is known. However, geometric location exhibits a flat structure and eventual usage may actually require a conversion to symbolic locations.

2) Virtual or Logical: Location attribute of a user can be represented using a virtual or logical construct. A computer IP address is one example of such a construct. Another example is an overlay ID for a host which is part of an overlay network. Figure 1 depicts a hierarchy of IP based logical grouping of computing devices. Devices in the logical group of 128.46.122.x have the same prefix and are separated by the last identifier. Similarly, going up the hierarchy, hosts having IP addresses starting with 128.46 are part of a group which is different from 128.47. It may be noted here that an IP based logical grouping of hosts creates virtual groups at a lower level.

B. Location Sensing

An overview of context discovery, especially location sensing is given in [18]. Sensing location context accurately and reliably is at the core of applying spatial constraints in the computing environment. A number of techniques have been reported and can be divided into two broad categories, namely: outdoors and indoors. The obvious choice for outdoor sensing of location is the Global Positioning System (GPS) [24]. GPS enables a cheap and accurate means of acquiring longitude, latitude and altitude by using time of flight and geometry of satellite transmitters as

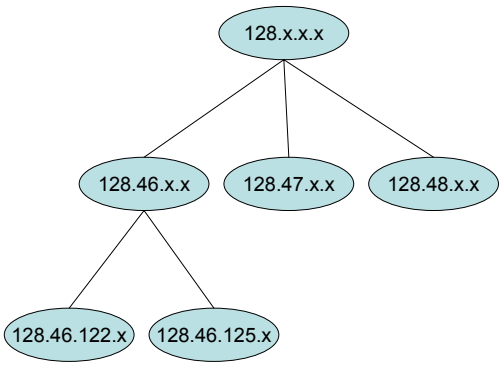

Fig. 1. An example of a virtual or logical location.

a basis of calculations. The accuracy of GPS has recently been further enhanced with the US Government turning off the degradation of the civilian data streams from the satellites.

Due to the fact that GPS signals can not be received indoors, a number of indoor location sensing techniques have been proposed. Notable among them the Olivetti Active Badge System [19], Xerox ParcTab [20] and the Cyberguide project [21].

Our current work builds on the location sensing research conducted by the community and assumes the availability of location contextual parameter with the desired resolution and representation. In this regard, the location acquisition and integration framework proposed by [18] serves as an appropriate underlying architecture for integration of location data from multiple heterogeneous sources.

C. Access Control Models

The NIST standard Role Based Access Control (RBAC) model proposed by [2] has four components: A set of users *Users*, a set of roles *Roles*, a set of permissions *Permissions* and a set of sessions *Sessions*. A role is abstracted as a set of permissions granted to users, which can be humans or software agents. Sessions are mechanisms to link roles with each other. A user can request the activation of a role if he is authorized to do so. Users, Roles, Permissions and Sessions have a number of functions defined in this model. The assignment of users to roles is achieved by the user role assignment (UA) function. Similarly, permissions are mapped to a role by a function called role permission assignment (PA). An important and useful feature of this

model is the definition of role hierarchy between roles whereby the $r_1 \geq r_2$ implies r_2 inheriting the permissions of r_1.

Generalized Temporal RBAC (GTRBAC) model [3] is the extension of RBAC that allows specification of a set of temporal constraints on role enabling and activation. Roles in GTRBAC have states associated with them which are *disabled*, *active* and *enabled*. The *disabled* state indicates that the role cannot be used in any user session. An *active* role signifies that authorized users can activate the role. Once the role is activated by a user, it said to be *enabled*.

IV. DEFINITIONS

User: A user is a human being or a physical computing device which can interact with the computing environment in any way. Examples of physical computing devices are software agents, network processors, embedded devices etc. We represent the set of all users by U. $u \in U$ is atomic in nature, by which we mean that a user cannot occupy multiple physical locations at the same time.

Physical Location: Physical location is defined as a symbolic or geometric representation of physical space which can be used for defining constraints on users. The set of all locations is denoted by L. $l_1 \in L$ is fully bounded and self contained space of N dimensions where $N \geq 2$. It may be noted here that certain locations can also be nonstationary, such as the inside of a vehicle or an airplane. Nonstationary locations become more relevant as access control decisions are made for users in an ambulance or a search and rescue aircraft which is not stationary.

Physical Symbolic Location: A physical location l is referred to as symbolic if it can be represented using symbols not having any relationship to its physical proximity or connection to any other location. Symbolic location can be grouped in and can have hierarchical relationship with other symbolic locations.

$\sqsubseteq (p)$ **Operator:** $\sqsubseteq$, is a hierarchy operator between symbolic locations and p is a topological relationship between symbolic locations forming hierarchical relationships. $L_1 \sqsubseteq (p) L_2$, where $L_1, L_2 \in L$ and $p \in P$, denotes that L_1 and L_2 have a topological relationship with each other which is represented by p. A hierarchical relationship between locations can also be multilevel. For example, $L_3 \sqsubseteq (p_b) L_2 \cup L_4 \sqsubseteq (p_c) L_2 \cup L_2 \sqsubseteq (p_a) L_1$ where $L_1, L_2, L_3, L_4 \in L$ and $p_a, p_b, p_c \in P$. p_a, p_b and p_c may be disjoint and the hierarchy may still hold. L_1 can also be a set of locations and $L_1 \sqsubseteq (p) L_2$ means that $\forall L_i \in L_1$ a hierarchical relationship exists between L_1 with L_2.

Elements of set P have been defined in the literature as the topological relations between spatial regions [22]. We represent the members of this set as $P = \{d, m, o, e, ct, i, cv, cB\}$, where each member stands for disjoint, meet, overlap, equal, contains, inside, covers and coveredBy, respectively (refer to [22] for precise definitions). These relations are mutually exclusive and any one of these relationships can hold for a given hierarchical relationship. An Example of symbolic location is $Cities \sqsubseteq (c)NewYork$, where $Cities = \{NewYorkCity, Albany, Chester\}$, and c refers to the topological relationship of containment.

Virtual Location: Virtual location v of a user implies his membership to a group defined in some computational sense. Example of a virtual group is the membership of a mobile computing device in an IP pool. Although the device may not be physically collocated with other members having IP addresses from the IP pool, the membership itself implies that all members are virtually collocated. It may be noted here that virtual locations may also exhibit hierarchical relationships amongst themselves. In order to represent virtual relationships between locations we extend the set P defined above and include v_m as a new member. v_m implies virtual membership of one virtual location in another virtual location. $G_1 \sqsubseteq (v_m)G_2$, where $G_1, G_2 \in v$ and $v_m \in P$, implies all members of G_1 are virtual members of G_2.

Figure 2 depicts the hierarchical relationship between symbolic locations and illustrates topological relationship between locations. In this example there is a contains (c) relationship between Washington DC and the Smithsonian Institute. Similarly, the relationship between Smithsonian Institute and Postal Museum is also contains (c), and so is between Smithsonian Institute and Picture Gallery. The topological relationship between Smithsonian Institute and National Zoo is that of overlap (o). It may be noted here that some parts of the zoo are physically located within the bounds of the institute and some are outside these bounds but these are still considered part of the institute. Formally, this topological relationship can be depicted as follows.
$(DC \sqsubseteq (c)SI) \cup (SI \sqsubseteq (c)PG) \cup (SI \sqsubseteq (c)PM) \cup (SI \sqsubseteq (o)NZ)$

V. The Generalized Spatio-Temporal Role Based Access Control Model

In order to exercise spatial access control of secure objects by applying spatial constraints in addition to temporal constraints, we propose the Generalized Spatio-Temporal RBAC (GSTRBAC) model which is the extension of the GTRBAC model for spatial constraint definition. In this regard the proposed model allows the specification of spatial constraints for roleactivation,

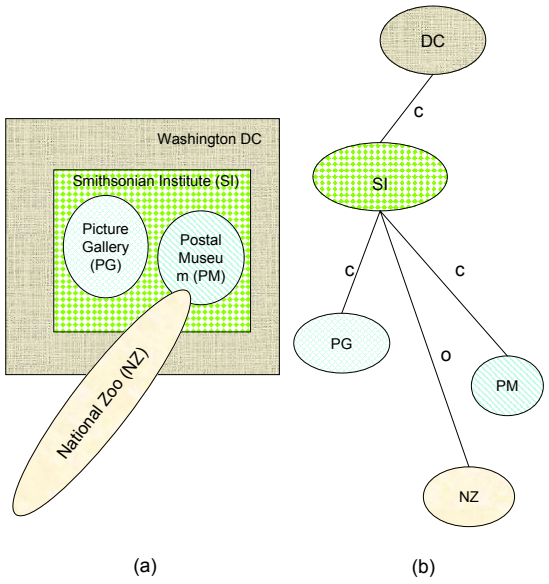

Fig. 2. hierarchical relationship between symbolic locations (a) Geographical relationship between Washington DC, Smithsonian Institute, Picture Gallery, Postal Museum and National Zoo. (b) hierarchical relationship between geographical entities along with topological relations.

roleenabling, roledisabling and rolepermission assignment, in addition to the existing temporal constraints in GTRBAC.

A. *Spatial Constraint Definition*

As pointed out in [3], role activation takes place when at least one user has activated a given role. There are situations when a role can only be activated if the user is in a specific location. Activation of roles based on spatial conditions can be achieved by defining spatial constraints. In order to model spatial constraints on such actions as role activation, userrole assignment and rolepermission assignment, we define spatial constraints as follows.

Formally, we define a spatial constraint spC as an expression of the form $< [L_1[\sqsubseteq (p)L_2][\cup] \ldots L_{n-1}[\sqsubseteq (p)L_n]], l_i >$, where

- $l_1 \ldots l_i, \ldots, l_n$ are symbolic locations $\in L$.
- $\sqsubseteq (p)$ is the hierarchical relationship between locations and p represents a member of the set P (defined in Section III).
- The constraint may be composed as a union between hierarchical relationships between multiple locations using the set $\cup$ operator.

The above defined spatial constraint expression may include a hierarchy of topologically related locations, complete with their relationship operators. Spatial constraints may be utilized for role enabling, spatial user-role assignment, spatial role-permission assignment and spatial activation. Location l_i refers to the spatial location at which the assigned action takes place and the defined hierarchy adds further resolution to the constraint. It may also be noted here that constraint expressions may be defined for the top level symbolic location in the hierarchy and topological relationships defined down the hierarchy. This ensures scalability in terms of role definition as new locations can be defined within the hierarchy without the need for role activation/deactivation/assignment/deassignment expressions for each constituent location in the hierarchy.

B. Spatial Constraints in GST-RBAC

GTRBAC enforces access control decisions by evaluating constraint expressions in the temporal domain [3]. The constraint expressions defined in GTRBAC are temporal constraints (which we denote by $TempC$) for role enabling, user-role and role-permission assignments, activation constraints for duration and cardinality, constraint enabling, run-time requests and triggers. To these temporal constraints we add spatial constraints namely, spatial constraint for role-enabling, spatial constraint for user-role assignment, spatial constraint for role-permission assignment and spatial activation constraint. Next we list these spatial constraints along with the constraint expressions and detailed explanation of each.

Spatial role enabling. The *enabled* state of a role signifies that any authorized user can activate the role. Spatial role enabling constraint defines the spatial relationship between locations for which the role can be enabled. A certain role may be enabled at a certain location, while it may not be enabled at other locations. Formally, we define a spatio-temporal role enabling expression as follows,

$([TempC], [spC], enable/disable, r)$ where $r \in R$.

The expression above has both the temporal constraint [3] ($TempC$) and spatial constraint (spC) as optional. In case one or both of these constraints is not defined in the expression, it is assumed that the role can be enabled without the relevant constraint. Once the role is enabled, any authenticated user can activate it.

Spatial user-role assignment. Assignment of a user to a role is performed by the user-role

assignment function. In case of spatial constraint spC, the user $u \in U$ is assigned to role $r \in r$ if the user satisfies a certain spatial constraint over locations $l_1 \ldots l_n$ for n locations. Formally, we define the spatio-temporal user-role assignment as,

$([TempC], [spC], assign_u/deassign_u, u, r)$ where $r \in R$, and $u \in U$.

Again, the temporal and spatial constraints are optional in this expression. A similar function is composed for each user in the policy base for assignment to as many roles as desired.

Spatial role-permission assignment. Assignment of permissions to roles is performed by the role-permission assignment function. When applied to spatial constraints, permissions are assigned to roles depending on the location. Formally we define the spatio-temporal role-permission assignment as

$([TempC], [spC], pr, assign_p/deassign_p, r)$ where $r \in R$, and $pr \in$ set of all permissions.

The permissions associated with a role depend on the location at which the role is being activated. This mechanism may be applied in situations where the access rights of users are restricted in some locales whereas are not restricted in others. For example, an employee accessing company records from his office may be presented with more details as compared to accessing the same records from his home.

Spatial activation. Users activate a role by sending an activation request. The request, along with other components like the user id, also contains the location parameter of the user. The spatial constraint defines the location at which the user can activate/deactivate the role. Activation of a role by a user may also be dictated by additional constraints such as temporal ones. We define the spatio-temporal activation constraint as

$([TempC], [spC], l_i, active_r/deactivate_r, u, r) \forall l_i \in L, r \in R, u \in U$.

The above expression allows a user to activate/deactivate a role based on temporal and spatial constraints. Since spC is the location constraint composed as a hierarchy of topological relationship between locations, we define l_i as the location from within the spC where the role activation actually takes place. Activation at other locations within spC is evaluated based on the topological relationships between locations (spC).

C. Spatial Separation of Duty Constraints (spSoD)

In order to curtail conflicts of interest in a role based system, RBAC offers Separation Of Duty (SoD), which allows definition of constraints for users to be authorized to activate roles.

SoD constraint does not allow a user to be activated to conflicting roles. The roles involved are mutually exclusive to the user. We extend this concept to include spatial constraints on SoD resulting in mutual exclusiveness of roles being dictated by the location of the user.

We define two types of SoD constraints, namely: spatially unrestricted SoD and spatially restricted SoD or spSoD. A spatially unrestricted SoD constraint remains unaffected by the location of the users and has the same semantics as defined by [2]. The spatially restricted SoD dictates spatial constraints on the application of SoD constraints on role activation.

A motivating example for a situation where spSoD is useful can be found in the health care domain. Two roles $Radiologist$ and $GeneralPractitioner$ cannot be activated by the same physician in the clinic. However, the same physician can activate both these roles simultaneously if he is in the ER of the hospital. The rational behind this is that in case of an emergency, the system should facilitate the attending physician fully, by allowing access to all possible information regarding a patient.

Formally we define spatial constraints on SoD as follows. $L_1, L_2 \in L, r_1, r_2 \in R$, such that $r_1 \neq r_2$, user $u \in U$, $(activateuser(u, r_1, L_1) \wedge \neg activateuser(u, r_2, L_1)) \vee (activateuser(u, r_2, L_1) \wedge \neg activateuser(u, r_1, L_1)) \wedge ((activateuser(u, r_1, L_2) \wedge activateuser(u, r_2, L_2))$. The above condition is expressed notationally as $spSoD(r_1, r_2, L_1) \wedge \neg spSoD(r_1, r_2, L_2)$ and can be generalized as $spSoD(r_1 \ldots r_n, L_1) \wedge \ldots \wedge [\neg] spSoD(r_1 \ldots r_n, L_m)$ for n roles and m locations. The spatial SoD in the afore mentioned motivating example can be expressed formally as,

$spSoD(Radiologist, GeneralPractitioner, Clinic)$ where $Clinic$ refers to the symbolic name of the location.

D. Spatial Role Hierarchy

RBAC offers role hierarchy as a means for roles to inherit permissions and users from each other. Role r_1 inherits role r_2 (expressed as $r_1 \geq r_2$)if all permissions of r_2 are also permissions of r_1. In this case r_1 acquires all users of r_2. Spatial constraints let us define spatial role hierarchies between roles activated from disparate physical locations. We define the following three categories of hierarchies.

1) *Spatially unrestricted role hierarchy.* Spatial constraints do not have any bearing on the semantics of role hierarchy and is similar to the one defined in [2].

Hierarchy type	Notation	Explanation
Disjoint	$r_{1(l_1)} \geq_d r_{2(l_2)} \forall r_1, r_2 \in R, l_1, l_2 \in L$	r_2 inherits permissions of r_1 iff $l_1 \sqsubseteq (d) l_2$
Meet	$r_{1(l_1)} \geq_m r_{2(l_2)} \forall r_1, r_2 \in R, l_1, l_2 \in L$	r_2 inherits permissions of r_1 iff $l_1 \sqsubseteq (m) l_2$
Overlap	$r_{1(l_1)} \geq_o r_{2(l_2)} \forall r_1, r_2 \in R, l_1, l_2 \in L$	r_2 inherits permissions of r_1 iff $l_1 \sqsubseteq (o) l_2$
Equal	$r_{1(l_1)} \geq_e r_{2(l_2)} \forall r_1, r_2 \in R, l_1, l_2 \in L$	r_2 inherits permissions of r_1 iff $l_1 \sqsubseteq (e) l_2$
Contains	$r_{1(l_1)} \geq_c r_{2(l_2)} \forall r_1, r_2 \in R, l_1, l_2 \in L$	r_2 inherits permissions of r_1 iff $l_1 \sqsubseteq (c) l_2$
Inside	$r_{1(l_1)} \geq_i r_{2(l_2)} \forall r_1, r_2 \in R, l_1, l_2 \in L$	r_2 inherits permissions of r_1 iff $l_1 \sqsubseteq (i) l_2$
Covers	$r_{1(l_1)} \geq_{cv} r_{2(l_2)} \forall r_1, r_2 \in R, l_1, l_2 \in L$	r_2 inherits permissions of r_1 iff $l_1 \sqsubseteq (cv) l_2$
CoveredBy	$r_{1(l_1)} \geq_{cB} r_{2(l_2)} \forall r_1, r_2 \in R, l_1, l_2 \in L$	r_2 inherits permissions of r_1 iff $l_1 \sqsubseteq (cB) l_2$

TABLE I

SPATIALLY RESTRICTED TOPOLOGICALLY RELATED ROLE HIERARCHY RELATIONS

2) *Simple spatially restricted role hierarchy* manifests itself in the form of a simple role hierarchy when a user activates a role from one location, while on the other hand, no hierarchy exists when the same role is activated from another location. Formally, $r_1, r_2 \in R$ $r_1 \geq_{l_1} r_2$ where $l_1 \in L$. This implies that when a user activates role r_1 while at location l_1, the permissions of role r_2 will be available to him. However, when the same user activates role r_1 from any location other than l_1, the hierarchical relationship does not hold. It may be noted here that there is no assumption about topological relationship between the two locations in question.
3) *Spatially restricted topologically related role hierarchy* occurs when two roles can be activated at two disjoint locations but can inherit each others permissions (role hierarchy holds) if the two corresponding locations have a certain topological relationship between each other. The various types of spatially restricted topologically related role hierarchy relations are defined in Table I.

VI. DISCUSSION

We now discuss access control implications of using spatial constraints for role activation/de-activation, assignment/deassignment, spatial separation of duty and hierarchy. During the life cycle of an access control policy, constraints are added and removed from the policy. The evolutionary nature of access control policies open the door for situations in which constraints

may conflict with each other. Our aim in this regard is to detect situations in which spatial constraints can be defined and used with harmful and detrimental consequences. In Section VIII we present a conflict resolution framework for detecting and correcting conflicting situations in the GSTRBAC policy. We divide our discussion into three broad categories, namely: missing topological relations, interacting spatial constraints and conflicting constraints.

A. Missing Topological Relations

In order to effectively discuss the effect of a missing topological relation in a spatial constraint we propose the following proposition.

Proposition 1: Let a location hierarchy be defined by the following topological relationship expression. $(L_1 \sqsubseteq (p_1)L_2) \cup (L_2 \sqsubseteq (p_2)L_3) \cup \ldots \cup (L_{n-1} \sqsubseteq (p_n)L_n)$. $p_1 \ldots p_n \in$ P. Let user u be located in location L_i where $i \in \{2 \ldots n\}$. Let role R be defined for activation/deactivation or assignment/deassignment at location L_1. Then user u can activate/deactivate or be assigned/de-assigned to role R if and only if location L_i is reachable from L_1 through topological relations.

We next illustrate the implications of Proposition 1. Consider locations $l_1, l_2, l_3, l_4, l_5 \in L$, roles $r_1, r_2, r_3 \in R$ and user $u_1 \in U$. The following spatial constraints are defined for spatial role activation. $((L_1 \sqsubseteq (c)L_2) \cup (L_2 \sqsubseteq (c)L_3) \cup (L_3 \sqsubseteq (c)L_4) \cup (L_3 \sqsubseteq (c)L_5), l_1, activate_r, u_1, r_1)$. Assume user u_1 to be at spatial location l_6. Since l_6 has not been defined in the location hierarchy, it cannot activate role r_1. This role activation scenario is depicted in Figure 3(a).

B. Interacting Spatial Constraints

As pointed out earlier, spatial constraints may interact with each other and present situations which may be violate access policy rules and intentions. Next we describe such a scenario in case of role activation with spatial constraints. It may be noted here that interacting spatial constraints may also have similar effects when used for role deactivation or assignment/deassignment.

The two spatial constraints depicted in Figure 3(b) and (c) are as follows. $((L_1 \sqsubseteq (c)L_2) \cup (L_2 \sqsubseteq (c)L_3) \cup (L_3 \sqsubseteq (c)L_5), l_1, activate_r, u_1, r_1)$ and $((L_5 \sqsubseteq (c)L_6), l_6, activate_r, u_1, r_2)$ respectively. As can be observed from the two constraints, r_1 can be activated by user being in any of the the four locations (l_1, l_2, l_3, l_5). Similarly, r_2 can be activated by the user assuming either location l_5 or l_6. Location l_5 is the common location for the two constraints and a user occupying this location activates both r_1 and r_2. Note that the inclusion of location l_5 in two

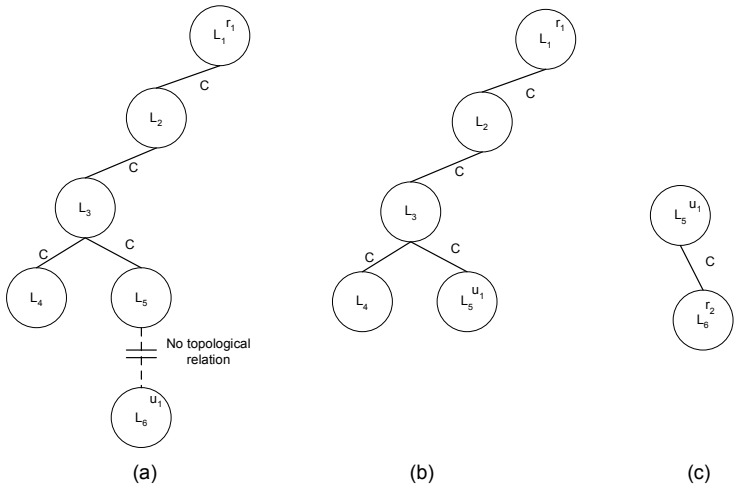

Fig. 3. Spatial hierarchical constraint for activation of roles. (a) Location l_6 not connected to location l_1, inhibiting u_1 to activate role r_1. (b) Spatial hierarchical constraint for activation of role r_1 by user u_1 at location l_5. (c) Spatial hierarchical constraint for activation of role r_2 by user u_1 at location l_5.

spatial constraints may be done on purpose or may also be inadvertent. This scenario uncovers the importance and criticality of composing spatial constriants.

C. Conflicting Constraints

We define and detect constraint conflicts which may arise in case of an interplay between various constraints. Three types of conflicts in GTRBAC have been defined in [3]. The first type is the conflict between events of the same category, also called Type 1 conflicts, which are conflicts because of the same pair of states of a role or assignment. The second type (Type 2) is the conflict between different categories. The third type of conflict (Type 3) [3] is the conflict between constraints. Type 3a conflict is for two constraints defined for role enabling or role assignment. Type 3b conflict can occur between the per-user activation constraint and the per-role activation constraint. We will elaborate Type 3 conflicts whereby we add spatial constraints to the already existing temporal constraints.

In order to incorporate spatial conflicts in the GTRBAC conflicts defined in [3], we extend the Type 3a conflicts to Type 3a(i), 3a(ii), 3a(iii).

- *Type 3a(i)*, are the role enabling or assignment conflicts as defined in [3], in which the two conflicting constraints are temporal in nature only.

- *Type 3a(ii)* conflict constraints are the role enabling and assignments conflicts in which both the conflicting constraints are spatial in nature. Consider $(Loc_1, l_1, l_2, l_3, enable, r_1)$ where $l_1, l_2, l_3 \in L, r_1 \in R$ and $(Loc_2, l_1, l_2, l_3, disable, r_1)$ where $l_1, l_2, l_3 \in L, r_1 \in R$ where $Loc_1 = Loc_2$. This implies that the same role r_1 will be enabled as well as disabled at the same location.

In addition, we define spatial hierarchical constraint conflict and designate it as Type 4. The Type 4 conflict will occur when there is a topological relationship conflict between the participating locations. $r_{1(l_1)} \geq_d r_{2(l_2)}$ conflicts with $r_{1(l_1)} \geq_c r_{2(l_2)}$ where $r_1, r_2 \in R, l_1, l_2 \in L$. Similarly, $r_{1(l_1)} \geq_d r_{2(l_2)}$ conflicts with $r_{1(l_1)} \geq_i r_{2(l_2)}$.

Next, we define the spatial SoD constraint conflict and designate it as Type 5. The Type 5 conflict occurs when a spSoD relationship is defined for the same location. An example is $spSoD(r_1, r_2, L_1) \wedge \neg spSoD(r_1, r_2, L_1)$ where $L_1 \in L$, and $r_1, r_2 \in R$ and $r_1 \neq r_2$.

Lastly, we define Type 6 constraint as a conflict between interacting spatial constraints and spatial SoD constraint. This type of conflict manifests itself when in addition to having interacting spatial constraints, we also have a spatial SoD constraint. As an example consider the interacting spatial constraint depicted in Figure 3(b) and (c), and add a spatial SoD constraint i.e. $spSoD(r_1, r_2, L_5)$. This constraint implies that a user may not activate roles r_1 and r_2 simultaneously at location L_5. The definition of this SoD causes the access control system to behave in an undecided manner.

VII. Example of Generalized Spatio-Temporal RBAC Policy

Next, we describe an example to illustrate the application and relevance of spatial constraints in practical environments. We will also use this example to illustrate spatio-temporal policy modeling in the subsequent sections. The example has been adapted from the one presented in [3].

We consider an access control policy for an Electronic Health Record (EHR), which has both temporal and spatial constraints. In order to illustrate composition of spatial constraints for the access control policy, we consider the layout of a floor in a hospital as depicted in Figure 4. We next compose spatial constraint expressions as shown in Table II. It may be noted here that constraint composition in the current example is a hierarchy of symbolic locations up to the county hospital and can be extended to the city and the state. This hierarchy allows for clear

definition of spatial context where there can be heterogeneity of symbolic locations (i.e more than one operation theaters in a hospital). The spatial constraint SpC_3, describes the relationship between the county hospital, surgery building and the second floor of the building. The second floor of the building is further related to the surgeon prep room, the recovery rooms and clinics. The relationships between each of these locations is also represented as part of the constraint. While making access control decisions based on SpC_3 constraint, a role defined for any one constituent location can be activated/deactivated or assigned/deassigned from all other locations part of SpC_3 constraint.

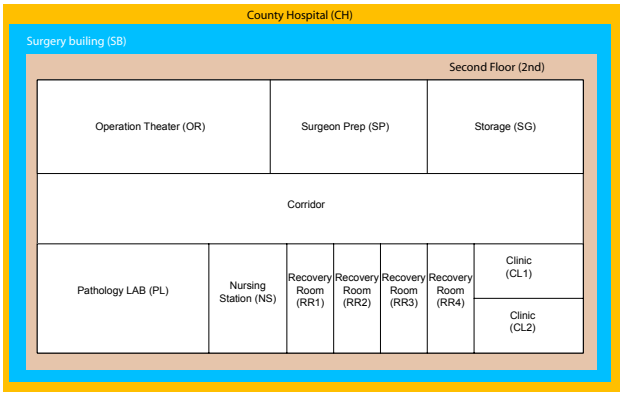

Fig. 4. Floor plan of a typical hospital

As described in [3], the periodicity constraint *DayTime* and *NightTime* are composed as follows.

$DayTime = ([1/1/2007, \infty], all.Days + 10.Hours \triangleright 12.Hours)$, and

$NightTime = ([1/1/2007, \infty], all.Days + 22.Hours \triangleright 12.Hours)$.

The roles in the access control policy and the relevant semantics are described in Table III. A portion of the access control policy exemplifying the application of spatial constraints in access control is depicted in Table IV. Row 2 of Table IV shows role `NightSurgeon` being enabled during temporal constraint $NightTime$ and at location described by spatial constraint SpC_1. Row 6 of Table IV signifies role `PrepSurgery` being enabled for location $SpC3$ if role `DaySurgeon` or `NightSurgeon` is enabled. This constraint implies that the role `PrepSurgery` can only be enabled if and only if one of the surgeon roles is enabled. Row 8 of the policy table implies user assignment to roles. User **Adam** is assigned to role `DaySurgeon`

Spatial Constraint	Notation	Explanation
SpC_1	$CH \sqsubseteq (c)SB \sqsubseteq (c)2nd \sqsubseteq (c)OR \sqsubseteq (m)SP$	Operation theater and surgeon prep area located on the second floor
SpC_2	$2nd \sqsubseteq (c)NS_1 \sqsubseteq (m)RR1 \sqsubseteq (m)RR2 \sqsubseteq (m)RR3 \sqsubseteq (m)RR4$	Nursing station and the recovery rooms on the second floor of the building
SpC_3	$(CH \sqsubseteq (c)SB \sqsubseteq (c)2nd \sqsubseteq (c)SP) \cup (2nd \sqsubseteq (c)RR1 \sqsubseteq (m)RR2 \sqsubseteq (m)RR3 \sqsubseteq (m)RR4) \cup (2nd \sqsubseteq (c)CL1 \sqsubseteq (m)CL2)$	Spatial location spanning the surgeon prep area, the recovery rooms and the two clinics
SpC_4	$CH \sqsubseteq (c)SB \sqsubseteq (c)2nd \sqsubseteq (c)((OR) \cup (NS \sqsubseteq (m)PL))$	Spatial location spanning the operation theater and nursing station which meets the pathology lab

TABLE II

SPATIAL CONSTRAINT EXPRESSIONS FOR GENERALIZED SPATIO-TEMPORAL RBAC POLICY FOR THE HEALTH CARE EHR.

within the confines of location constriant SpC_1, during *DayTime*. User **Beth** is assigned the role `SeniorNurse` while at location SpC_2 any time of the day. User **Ami** is assigned the role `NightNurse` only when she is within the confines of Nursing Station (NS) and at time *NightTime*. However, **Meg** is assigned role `NightNurse` at all locations represented by SpC_2 and at *NightTime*. It may be noted here that although **Meg** and **Ami** are being assigned the same role (i.e. `NightNurse`) and during the same time of the day, but **Ami** can only access permissions of this role within the nursing station, while **Meg** can access the same permissions from NS as well as the recovery rooms. In rows 9 and 10 of Table IV, we present activation constraints of activation/deactivation of users **Adam** and **Mark**. Both the users are activated when they are detected to be within the confines of SpC_1 and at specified times of the day. This serves to restrict the roles active at any one time in any one location. Row 11 of Table IV defines spatial SoD between roles `SeniorNurse` and `NightNurse` in location NS_1. A user can be assigned the two roles simultaneously, at all locations other than NS_1. The rational for this constraint is that in the presence of a senior nurse the junior nurse should not be able to acquire permissions while at location NS_1. However, the two roles can be activated by the same user in locations other than NS_1 but within the spatial definition SpC_2. Row 13 shows the activation constraint for users **Ami** and **Meg** for the locations specified in the assignment constraint above.

Role Name	Semantics
`DaySurgeon`	The role is enabled and activated at $DayTime$, every day of the week and for locations inside Operation Room and the Surgeon Prep (represented by SpC_1 constraint)
`NightSurgeon`	The role is enabled and activated at $NightTime$, every day of the week and for locations inside Operation Room and the Surgeon Prep (represented by SpC_1 constraint)
`SeniorNurse`	Role enabled and activated all day in locations represented by spatial constraint SpC_2
`NightNurse`	Role enabled and activated in $NightTime$ at locations specified by SpC_2
`TechnicianSurgery`	Role enabled and activated all day for five working days of the week at locations represented by spatial constraint SpC_4
`PrepSurgery`	Role enabled and activated all day at locations specified by SpC_3 and only if roles `DaySurgeon` and `NightSurgeon` are enabled
`SurgeryLab`	Role enabled and activated all day at locations specified by SpC_3

TABLE III

ROLE DEFINITION AND SEMANTICS FOR GENERALIZED SPATIO-TEMPORAL ACCESS POLICY OF HEALTH CARE EHR

Row 14 of Table IV depicts the spatial hierarchical relationship between roles `SeniorNurse` and `NightNurse` at location NS_1. The permissions of role `NightNurse` are inherited by the `SeniorNurse` only in location NS_1. This and the spSoD constraint defined in Row 11 demonstrate how permissions can be made available to a certain role in a certain location while maintaining a separation of conflicts.

VIII. SPECIFICATION MODELING AND CONFLICT RESOLUTION OF GENERALIZED SPATIO-TEMPORAL ROLE BASED ACCESS CONTROL

In this section we develop a specification model for GSTRBAC policy and outline methodology for formal conflict resolution of access control policies using lightweight formalism.

1	($DayTime$, SpC_1, *enable*, DaySurgeon)
2	($NightTime$, SpC_1, *enable*, NightSurgeon)
3	(SpC_2, *enable*, SeniorNurse)
4	($NightTime$, SpC_2, *enable*, NightNurse)
5	(DayTime, SpC_4, *enable*, TechnicianSurgery)
6	(*enable*(DaySurgeon) $\vee$*enable*(DaySurgeon))$\rightarrow$(SpC_3, enable, PrepSurgery)
7	(SpC_3, enable, SurgeryLab)
8	($DayTime$, SpC_1, $assign_u$, Adam, DaySurgeon); ($NightTime$, SpC_1,$assign_u$, Mark, NightSurgeon); (SpC_2, $assign_u$, Beth, SeniorNurse); ($NightTime$, NS, $assign_u$, Ami, NightNurse); ($NightTime$, SpC_2, $assign_u$, Meg, NightNurse); (PL, $assign_u$, Andrew, TechnicianSurgery); ($assign_u$, Kevin, PrepSurgery); ($assign_u$, Bill, SurgeryLab)
9	($Daytime$, SpC_1, OR, $activate_r$, Adam, $DaySurgeon$)$\rightarrow$($Nighttime$, SpC_1, OR, $deactivate_r$, Mark, $NightSurgeon$)
10	($Nighttime$, SpC_1, OR, $activate_r$, Mark, $NightSurgeon$)$\rightarrow$($Daytime$, SpC_1, OR, $deactivate_r$, Adam, $DaySurgeon$)
11	$spSoD$(SeniorNurse, NightNurse, NS_1)
12	(SpC_2, NS_1, $activate_r$, Beth, $SeniorNurse$)
13	($NightTime$, NS_1, $activate_r$, Ami, $NightNurse$); ($NightTime$, SpC_2, NS_1, $activate_r$, Meg, $NightNurse$)
14	$SeniorNurse \geq_{NS_1} NightNurse$
15	$SeniorNurse \geq_{NS_1} SurgeryLab$

TABLE IV

FRAGMENT OF ACCESS CONTROL POLICY FOR GENERALIZED SPATIO-TEMPORAL ACCESS POLICY OF EHR

Our modeling activity in this regard is two tiered. Firstly, we develop a GSTRBAC policy specification model by capturing the defined GSTRBAC features in a light-weight formal model, with the hope to uncover potential conflicts in GSTRBAC specifications. We use the Alloy [25] specification language and the accompanying constraint analyzer to verify the specification model. Next, we utilize the specification model created thus far to develop model of GST-RBAC policy for conflict analysis and resolution. The aim in this regard is to provide a formal framework to the policy administrator to formally compose GSTRBAC policies and verify policy composition prior to actually implementing it. This methodology also allows conflict free evolution of both the access control model being used to implement security (in this case GSTRBAC) and the actual policy instance being implemented in the organization.

A. Overview of Alloy

Alloy is a predicate logic and set theoretic based first-order modeling notation used to model software components. It has a number of features which make it convenient for formal specification of access control policies. Some of the pertinent ones are mentioned below.

- Software models created using the alloy formalism capture the structure of the original software rather than the events.
- Alloy models are declarative, implying the model lists a system's properties and constraints [26]. Alloy declarations specify conditions and constraints which cause software to go from one state to another.
- Once the software is specified using alloy formalism it can be analyzed using the accompanying constraint analyzer which allows simulation of the model to generate structure and behavior in the form of examples of the system. The constraint analyzer also allows checking of models using a counter example approach which pinpoints model properties which do not hold under the specified conditions.

Access control policies are inherently declarative in nature and exhibit a structure which is held together using a set of constraints and assertions. Policy assertions cause the security system to go from one state to the next which can be achieved only if the constraints so allow. Since access control policies may evolve over time, the need for conflict resolution cannot be overstated. Creating a policy specification model using Alloy and employing the Alloy constraint analyzer helps in composition and evolution of consistent access control policies.

Next, we outline the basic components of Alloy as discussed in [27]. For an exhaustive discussion refer to [15].

Atom An atom is the basic building block of the Alloy model. It is indivisible, immutable and uninterpreted. All relations are composed of atoms.

Relations A relation establishes relationship between atoms. A relation is a collection of tuples which in turn are made up of atoms. A scalar is also represented as a singleton relation. The structure of the software model is composed using relations.

Operators Operators in Alloy are used to compose more expressive expressions. All expressions denote relations. Union, intersection and difference over relations have the same semantics as in set theory and are denoted by $+$, $\&$ and $-$, respectively. Another important operator is

the composition or join denoted by (.). p.q results in a relation by taking every combination of a tuple of p and a tuple of q, and including their join if it exists.

Formulas Larger formulas can be formed by combining smaller formulas using logical operators. These operators are && (and), || or, and ! (not). in and $=$ operators are used for subset and set equality. $->$ implies implication. Alloy also allows quantifiers for formulas which are some, all, no and sole. some $x|F$ means these is some x for which F holds. Similarly, all implies all values of x, no implies no value of x, and sole implies at the most one value of x which satisfies F.

A brief explanation of the underlying working of Alloy Analyzer is available at [28] and we restate it here. "The Alloy Analyzer is essentially a compiler. It translates the problem to be analyzed into a (usually huge) boolean formula. This formula is handed to a SAT solver, and the solution is translated back by the Alloy Analyzer into the language of the model. All problems are solved within a userspecified scope that bounds the size of the domains, and thus makes the problem finite (and reducable to a boolean formula)".

B. GST-RBAC Policy Specification Modeling using Alloy

In order to capture the structure of the GSTRBAC policy, we define objects as the basic pillars of the policy, together with constraints defining rules which govern the interplay between these objects. The complete Alloy model is attached as Appendix I. The model is divided into four basic parts, namely: declarations, invariants, functions/predicates and assertions.

```
sig User {}
sig Role{}
sig Permission{}
sig Location {operator: SpatialOperators -> Location}
sig SpatialOperators{}
sig Time {}

sig RoleEnable {re_member :some Role->some Location ->some Time}
sig RoleDisable {rd_member : some Role->some Location ->some Time}
sig UserRoleAssignment {URA_member : some User ->some Role -> some Location ->some Time}
sig UserRoleDeAssignment {URDA_member :some User ->some Role ->some Location ->some Time}
sig RolePermissionAssignment {RPA_member :some Role->some Permission  ->some Location ->some Time}
sig RolePermissionDeAssignment {RPDA_member :some Role->some Permission ->some Location ->some Time}
sig UserRoleActivation {URAct_member :some User-> some Role->some Location->some Time}
sig UserRoleDeActivation {URDAct_member :some User->some Role-> some Location->some Time }
sig RoleHierarchy {rh_member: some Role -> some Role -> some Location -> some Time}
```

Fig. 5. Signature declarations of GSTRBAC policy

Declarations of objects is in the form of *sig* structure and includes elements such as fields (see Figure 5). The sig *User*, sig *Role*, sig *Permission*, sig *Location* and sig *Time* are the five basic signatures of objects of the model. It may be noted here that User, Role and Permission signify the usual components of the RBAC model. Location and Time are the two context parameters which are used for all access control decisions. The next two signatures are,

```
sig RoleEnable {
re_member :some Role->some Location ->some Time}
sig RoleDisable {
rd_member : some Role->some Location ->some Time}
```

The *RoleEnable* signature has a field re_member that maps *Roles* to *Location* to *Time*. In fact re_member is a four-way mapping associating *RoleEnable*, *Role*, *Location* and *Time*. re_member can be thought of as a relation which represents the roles which have been enabled for a particular location and time. Signature *RoleDisable* has similar dynamics as the *RoleEnable*, only difference being that it refers to the set of roles which have been disabled for the respective times and locations, referred to by the sole attribute rd_member.

In order to constrain the structure of the access control policy, facts have been defined. Most facts in the alloy model of GSTRBAC correspond to the conflicts defined in [3] and in Section VI of this paper. The constraint *UsersEnableNotDisable* is defined as follows:

```
fact UsersEnableNotDisable{//conflict type 1a
some rd: RoleDisable, re: RoleEnable,
r: Role, t: Time, l: Location |
((r -> l-> t) in re.re_member =>
(r -> l-> t) not in rd.rd_member &&
(r -> l-> t) not in re.re_member =>
(r -> l-> t) in rd.rd_member)
}
```

The fact *UsersEnableNotDisable* ensures that the if a role has been enabled for a location l and time t, then it will not be disabled. Conversely, if a role has been disabled at a location l and time t, then it will not be enabled. The two signature structures, *RoleEnable* and *RoleDisable* are utilized for maintaining the two disjoint sets. This fact is semantically equivalent to conflict Type 1a defined in [3] and the spatial role enabling conflict defined in Section VI.

In order to enable a role in our model, we write a predicate, *EnableRole*. The text of the predicate is as follows:

```
pred EnableRole (rd, rd': RoleDisable,
re, re': RoleEnable, r: Role, t: Time, l: Location) {
re'.re_member=re.re_member+ (r-> l -> t) &&
rd'.rd_member=rd.rd_member- (r-> l -> t)
}
```

In predicate *EnableRole*, re and re' are the role enabled sets before and after the role enabling action. The difference between the two sets is the addition of the current tuple of role, location, time. Additionally, rd and rd' are the two sets which represent the disabled roles before and after the role enabling action. Similarly, the *DisableRole* predicate (see Appendix I) works opposite to the enable role predicate.

In order to illustrate the role enabling action, we execute the following command,

```
run EnableRole for 3 but 2 RoleEnable, 2 RoleDisable
```

Executing the above predicate causes Alloy to look for examples where the facts and the predicate are true. In case it cannot find such an example, alloy returns with a report of inconsistency. In our current case it comes up with an example where the fact *UsersEnableNotDisable* and predicate *EnableRole* are true. It is possible to display this example using Alloy in a number of formats, including visual, as a tree, a table or XML. We feel that for smaller examples a visual representation is the best, however as the size of the example grows, a table or a tree view is more convenient. The visual representation of role enabling is depicted in Figure 6. It is noted that role Role0 has been enabled for location Location0 and at two times Time0 and Time2. On the other hand the same role, Role0 is disabled at the same location Location0 but at time Time1, which conforms to the defined fact. The above run command executes the predicate and looks for examples where the facts (constraints) hold and the predicate evaluates to true.

In order to evaluate the complete working of the *EnableRole* and *DisableRole* pair, we create and execute an assertion which is as follows,

```
assert CheckRoleEnable {
all rd, rd', rd'': RoleDisable, re, re', re'': RoleEnable,
r: Role, t: Time, l: Location |
no r.(re.re_member) and
EnableRole (rd, rd', re, re', r, t, l)
and DisableRole (rd', rd'', re', re'', r, t, l)
implies re.re_member=re''.re_member
```

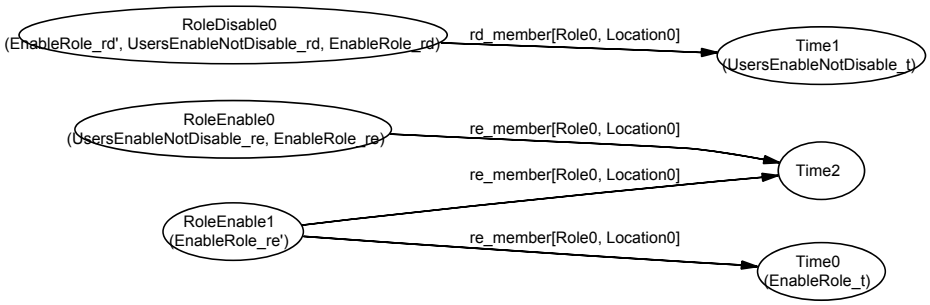

Fig. 6. Role enable/disable for one role at one location and at three times

```
}
check CheckRoleEnable for 2
```

The assertion *CheckRoleEnable* enables a role and then disables it. Finally, it checks the contents of the role enable structure before and after and finds the two to be equivalent. This assertion looks for counterexamples whereby facts and the assertion are at odds with each other. In our case, no counter example is found, implying that no violations of the conflict 1a (as defined in [3] and Section VI) were found for two instances of role enabling.

The next pair of signatures are the *UserRoleAssignment* and *UserRoleDeAssignment*. In line with *RoleEnable* signature, *UserRoleAssignment* refers to the set of users which have been assigned roles for certain locations and times. The *UserRoleDeAssignment* is the set of users, roles, locations and times which have be deassigned.

We represent conflict Types 1b and 2 (as defined in [3] and Section VI) as facts *UsersAssigned-NotDeassigned* and *RoleEnabledThenAssigned*, listed in Appendix I. In order to incorporate conflict Type 1b in the model, the fact *UsersAssignedNotDeassigned* ensures that if a user is assigned to a role at a specific location and time, then the same user cannot also exist as a deassigned user of a role for the same location and time. This is made possible by the following alloy formula:

```
((u->r -> l-> t) in ura.URA_member =>
(u->r -> l-> t) not in urda.URDA_member &&
(u->r -> l-> t) not in ura.URA_member =>
(u->r -> l-> t) in urda.URDA_member)
```

ura and *urda* are instances of the signatures *UserRoleAssignment* and *UserRoleDeAssignment*. Conflict Type 2 has been incorporated in the model as fact *UsersAssignedNotDeassigned* which ensures that a user can only be assigned to an enabled role. We assign users with the help of predicate *UserRoleAssignPred*. Relevant portion of the output generated Alloy is shown in Figure 7.

UserRoleAssignment

policy/UserRoleAssignment_0,

URA_member : (policy/User) ->some ((policy/Role) ->some ((policy/Location) ->some (policy/Time)))

policy/UserRoleAssignment_0	policy/User_0	policy/Role_0	policy/Location_0	policy/Time_0
policy/UserRoleAssignment_0	policy/User_0	policy/Role_0	policy/Location_1	policy/Time_0
policy/UserRoleAssignment_0	policy/User_0	policy/Role_1	policy/Location_0	policy/Time_0
policy/UserRoleAssignment_0	policy/User_0	policy/Role_1	policy/Location_1	policy/Time_0

UserRoleDeAssignment

policy/UserRoleDeAssignment_0,

URDA_member : (policy/User) ->some ((policy/Role) ->some ((policy/Location) ->some (policy/Time)))

policy/UserRoleDeAssignment_0	policy/User_0	policy/Role_0	policy/Location_0	policy/Time_1
policy/UserRoleDeAssignment_0	policy/User_0	policy/Role_0	policy/Location_1	policy/Time_1
policy/UserRoleDeAssignment_0	policy/User_0	policy/Role_1	policy/Location_0	policy/Time_1
policy/UserRoleDeAssignment_0	policy/User_0	policy/Role_1	policy/Location_1	policy/Time_1

Fig. 7. Partial Alloy output after running predicate *UserRoleAssignPred*

Note that user User_0 has been assigned to roles Role_0 and Role_1 at two locations (Location_0, Location_1) and times (Time_0, Time_1). The tables depicting UserRoleAssignment and UserRoleDeAssignment are both disjoint because of fact *UsersAssignedNotDeassigned*.

Similar analysis is done for modeling of conflicts between events of role activation and role deactivation (Conflict 1d in [3] and the spatial role activation conflict, Type3a(ii), defined in Section VI). We use the signatures *UserRoleActivation* and *UserRoleDeActivation* which correspond to the set of activated users and the set of deactivated users. The conflict has been

represented by the fact *UsersActivatedNotDeActivated*.

The predicate used to test this conflict is *UserRoleActivationPred*. Part of the example generated by Alloy is depicted in Figure 8. As may be noted, user User_0 has been assigned to roles Role_0 and Role_1 at two locations (Location_0, Location_1) and times (Time_0, Time_1) and the two sets *UserRoleActivation* and *UserRoleDeActivation* have disjoint tuples. It may be noted that the Alloy outputs shown in Figure 7 and Figure 8 are are not the same runs and may not correlate with each other.

UserRoleActivation

policy/UserRoleActivation_0,

URAct_member : (policy/User) -> ((policy/Role) -> ((policy/Location) -> (policy/Time)))

policy/UserRoleActivation_0	policy/User_0	policy/Role_0	policy/Location_0	policy/Time_0
policy/UserRoleActivation_0	policy/User_0	policy/Role_0	policy/Location_0	policy/Time_1
policy/UserRoleActivation_0	policy/User_0	policy/Role_1	policy/Location_0	policy/Time_0
policy/UserRoleActivation_0	policy/User_0	policy/Role_1	policy/Location_1	policy/Time_0

UserRoleDeActivation

policy/UserRoleDeActivation_0,

URDAct_member : (policy/User) -> ((policy/Role) -> ((policy/Location) -> (policy/Time)))

policy/UserRoleActivation_0	policy/User_0	policy/Role_0	policy/Location_1	policy/Time_0
policy/UserRoleActivation_0	policy/User_0	policy/Role_0	policy/Location_1	policy/Time_1
policy/UserRoleActivation_0	policy/User_0	policy/Role_1	policy/Location_0	policy/Time_1
policy/UserRoleActivation_0	policy/User_0	policy/Role_1	policy/Location_1	policy/Time_1

Fig. 8. Relevant Alloy output after running predicate *UserRoleActivationPred*

Conflict between events of different types is also encoded within the alloy GSTRBAC policy model (Type 3(i) and (ii) as defined in [3] and Section VI). The two facts which represent these conflicts are *RoleEnabledThenAssigned* and *RoleAssignedThenActivated* and are listed in Appendix I. *RoleAssignedThenActivated* ensures that the user is only assigned to a role which

has been enabled. Similarly, *RoleAssignedThenActivated* ensures that a user can only activate a role from a certain location and time which has been assigned for the specified location and time. Relevant output generated by Alloy is depicted in Figure 9.

UserRoleAssignment

policy/UserRoleAssignment_0,

URA_member : (policy/User) -> ((policy/Role) -> ((policy/Location) -> (policy/Time)))

policy/UserRoleAssignment_0	policy/User_0	policy/Role_0	policy/Location_0	policy/Time_0
policy/UserRoleAssignment_0	policy/User_0	policy/Role_0	policy/Location_0	policy/Time_1
policy/UserRoleAssignment_0	policy/User_0	policy/Role_1	policy/Location_0	policy/Time_0
policy/UserRoleAssignment_0	policy/User_0	policy/Role_1	policy/Location_1	policy/Time_0

UserRoleDeAssignment

URDA_member : (policy/User) -> ((policy/Role) -> ((policy/Location) -> (policy/Time)))

policy/UserRoleAssignment_0	policy/User_0	policy/Role_0	policy/Location_1	policy/Time_0
policy/UserRoleAssignment_0	policy/User_0	policy/Role_0	policy/Location_1	policy/Time_1
policy/UserRoleAssignment_0	policy/User_0	policy/Role_1	policy/Location_0	policy/Time_1
policy/UserRoleAssignment_0	policy/User_0	policy/Role_1	policy/Location_1	policy/Time_1

UserRoleActivation

policy/UserRoleActivation_0,

URAct_member : (policy/User) -> ((policy/Role) -> ((policy/Location) -> (policy/Time)))

policy/UserRoleActivation_0	policy/User_0	policy/Role_0	policy/Location_0	policy/Time_0
policy/UserRoleActivation_0	policy/User_0	policy/Role_0	policy/Location_0	policy/Time_1
policy/UserRoleActivation_0	policy/User_0	policy/Role_1	policy/Location_0	policy/Time_0
policy/UserRoleActivation_0	policy/User_0	policy/Role_1	policy/Location_1	policy/Time_0

UserRoleDeActivation

policy/UserRoleDeActivation_0,

URDAct_member : (policy/User) -> ((policy/Role) -> ((policy/Location) -> (policy/Time)))

policy/UserRoleActivation_0	policy/User_0	policy/Role_0	policy/Location_1	policy/Time_0
policy/UserRoleActivation_0	policy/User_0	policy/Role_0	policy/Location_1	policy/Time_1
policy/UserRoleActivation_0	policy/User_0	policy/Role_1	policy/Location_0	policy/Time_1
policy/UserRoleActivation_0	policy/User_0	policy/Role_1	policy/Location_1	policy/Time_1

Fig. 9. Relevant Alloy output after running predicate *UserRoleActivationPred*

The definition of spSoD (as formally treated in Section VI) in the alloy specification model of GSTRBAC model is achieved by adding the following predicate to the model,

```
pred SoD (r1, r2:Role, u:User, uract: UserRoleActivation,
```

```
l:Location, t: Time){
(u->r1-> l-> t) in (uract.URAct_member) =>
(u->r2-> l-> t) not in (uract.URAct_member)
}
run SoD for 2
```

This predicate ensures that spSoD can be defined between roles r1 and r2 at location l and time t. The signature object used in this case is *UserRoleActivation*. A user u can activate role r1 at location l and at time t but cannot activate role r2 at the same time and location. spSoD has been defined as a predicate so that it can be called during composition of the policy. Note that this predicate can also be used to define time-based separation of duty constraints as defined in [3].

Spatial role hierarchy, as defined in Section VI is captured in the GSTRBAC Alloy specification model by the signature *RoleHierarchy*, shown below,

```
sig RoleHierarchy
{rh_member :textit{Role} -> Role -> Location -> Time}
```

The attribute rh_member is a five way mapping between role hierarchy, senior role, junior role, location and time. The addition of time parameter in this mapping attribute ensures that this structure can be utilized to define temporal role hierarchies proposed in [3]. The fact *OneWayHierarchy* ensures that if there is a role hierarchy defined between two roles r1 and r2, then it cannot be defined in the reverse sense (r2 to r1). Note that this relationship is constrained in the temporal and spatial dimension, implying that the direction of a hierarchy is preserved for a given time and location.

```
fact OneWayHierarchy{
some rh: RoleHierarchy,
u: User, r1, r2: Role, t: Time, l: Location|
((r1->r2 -> l-> t) in rh.rh_member =>
(r2->r1 -> l-> t) not in rh.rh_member )
}
```

In the above discourse, we have introduced a methodology for the creation of the GSTRBAC specification model using Alloy. This has been achieved by representing conflicts as facts and ensuring that each fact can be validated for the signatures depicted in Figure 5. Complete listing of the GSTRBAC specification model is placed as Appendix I.

C. GST-RBAC Access Control Policy Modeling using Alloy

In the previous subsection we developed the GSTRBAC specification model using the Alloy framework. In this subsection we use the policy framework created thus far to model an access control policy and illustrate the conflict resolution mechanism afforded by Alloy to create conflict free policy artifact which can be composed and analyzed piecemeal. In the following discussion we demonstrate the access control policy using example from Section VII. In favor of brevity and to demonstrate out approach, we describe in detail roleenabling, userroleassignment, userroleactivation and spatial SoD functions of the GSTRBAC policy.

The complete listing of the policy specification for role enabling is placed at Appendix II. Role signature is extended by DaySurgeon, Time signature is extended by DayTime and NightTime, and Location signature is extended by SpC1. The policy assertions stated in Rows 1 and 3 of Table IV are represented in the Alloy model as facts shown below,

```
fact DaySurgeonEnableAtDayTime{
some rd: RoleDisable, re: RoleEnable,
r: DaySurgeon, t: DayTime, l: SpC1 |
(r -> l-> t) in re.re_member
}
fact DaySurgeonDisableAtNightTime{
some rd: RoleDisable, re: RoleEnable,
r: DaySurgeon, t: NightTime, l: SpC1 |
(r -> l-> t) in rd.rd_member
}
fact NightSurgeonEnableAtNightTime{
some rd: RoleDisable, re: RoleEnable,
r: NightSurgeon , t: NightTime, l: SpC1 |
(r -> l-> t) in re.re_member
}
fact NightSurgeonDisableAtDayTime{
some rd: RoleDisable, re: RoleEnable,
r: NightSurgeon, t: DayTime, l: SpC1 |
(r -> l-> t) in rd.rd_member
}
```

The facts *DaySurgeonEnableAtDayTime* and *NightSurgeonEnableAtNightTime* enable the role *DaySurgeon* and *NightSurgeon* at *DayTime* and *NightTime*, respectively. We next run the predicate *EnableRole* and observe the output, a partial view of which is depicted in Figure 10. Since the predicate evaluates to true and no fact is violated, Alloy outputs an example. As can

be seen in this output, the role *DaySurgeon_0* is enabled at Daytime and at location SpC1. On the other hand the role *DaySurgeon_0* is disabled at NightTime at location SpC1. Next, we create a fact *NightSurgeonDisableAtNightTime* which violates the previously defined fact *NightSurgeonEnableAtNightTime*, as follows,

```
fact NightSurgeonDisableAtNightTime{  //conflict
some rd: RoleDisable, re: RoleEnable,
r: NightSurgeon, t: NightTime, l: SpC1 |
(r -> l-> t) in rd.rd_member
}
```

As expected Alloy does not return an example for the role enable and disable signatures and states that the model is inconsistent.

RoleEnable

policy/RoleEnable_0,

re_member : (policy/Role) -> ((policy/Location) -> (policy/Time))

policy/RoleEnable_0	policy/DaySurgeon_0	policy/SpC1_0	policy/DayTime_0
policy/RoleEnable_0	policy/NightSurgeon_0	policy/SpC1_0	policy/NightTime_0

RoleDisable

policy/RoleDisable_0,

rd_member : (policy/Role) -> ((policy/Location) -> (policy/Time))

policy/RoleDisable_0	policy/DaySurgeon_0	policy/SpC1_0	policy/NightTime_0
policy/RoleDisable_0	policy/NightSurgeon_0	policy/SpC1_0	policy/DayTime_0

Fig. 10. Relevant Alloy output after running predicate *EnableRole* for the EHR example in Section VII

We now model the first two assignment functions from Row 8 of Table IV. Two users, Adam and Mark are being assigned to roles *DaySurgeon* and *NightSurgeon*, respectively. The constraints on the two assignments are both temporal as well as spatial. We extend the *User* signature for Adam and Mark and define four new facts to represent the assignment function, as follows. Complete Alloy listing of the model is placed as Appendix III.

```
fact AdamAssignedToDaySurgeonAtDayTimeAtSpC1{
```

```
some ura: UserRoleAssignment,
urda: UserRoleDeAssignment, u: Adam,
r: DaySurgeon, t: DayTime, l: SpC1 |
(u->r -> l-> t) in ura.URA_member and
(u->r -> l-> t) not in urda.URDA_member
}

fact AdamNotAssignedToDaySurgeonAtNightTimeAtSpC1{
some ura: UserRoleAssignment,
urda: UserRoleDeAssignment, u: Adam,
r: DaySurgeon, t: NightTime, l: SpC1 |
(u->r -> l-> t) not in ura.URA_member and
(u->r -> l-> t) in urda.URDA_member
}

fact MarkAssignedToNightSurgeonAtNightTimeAtSpC1{
some ura: UserRoleAssignment,
urda: UserRoleDeAssignment, u: Mark,
r: NightSurgeon, t: NightTime, l: SpC1 |
(u->r -> l-> t) in ura.URA_member and
(u->r -> l-> t) not in urda.URDA_member
}

fact MarkNotAssignedToNightSurgeonAtDayTimeAtSpC1{
some ura: UserRoleAssignment,
urda: UserRoleDeAssignment, u: Mark,
r: NightSurgeon, t: DayTime, l: SpC1 |
(u->r -> l-> t) not in ura.URA_member and
(u->r -> l-> t) in urda.URDA_member
}
```

The fact *AdamAssignedToDaySurgeonAtDayTimeAtSpC1* ensure that Adam is assigned to the role DaySurgeon at time DayTime and location SpC1. similarly, the fact and *MarkAssignedToNightSurgeonAtNightTimeAtSpC1* does a similar assignment for Mark at a different time, but the same location. The facts *AdamNotAssignedToDaySurgeonAtNightTimeAtSpC1* and *MarkNotAssignedToNightSurgeonAtDayTimeAtSpC1* ensure that Adam and mark are not assigned to the wrong roles at the wrong times.

We run the predicate *UserRoleAssignPred* and Alloy returns an example for the policy which

is depicted in Figure 11. Note that users Adam and Mark have been made part of the *UserRoleAssignment* signature structure for role assignment at designated times and locations. On the other hand they have been made part of the *UserRoleDeAssignment* signature structure for the specified roles, time and locations.

UserRoleAssignment

policy/UserRoleAssignment_0, policy/UserRoleAssignment_1,

URA_member : (policy/User) -> ((policy/Role) -> ((policy/Location) -> (policy/Time)))

policy/UserRoleAssignment_0	policy/Adam_0	policy/DaySurgeon_0	policy/SpC1_0	policy/DayTime_0
policy/UserRoleAssignment_0	policy/Mark_1	policy/NightSurgeon_0	policy/SpC1_0	policy/NightTime_1
policy/UserRoleAssignment_1	policy/Adam_0	policy/DaySurgeon_0	policy/SpC1_0	policy/DayTime_0
policy/UserRoleAssignment_1	policy/Mark_1	policy/NightSurgeon_0	policy/SpC1_0	policy/NightTime_1

UserRoleDeAssignment

policy/UserRoleDeAssignment_0, policy/UserRoleDeAssignment_1,

URDA_member : (policy/User) -> ((policy/Role) -> ((policy/Location) -> (policy/Time)))

policy/UserRoleDeAssignment_1	policy/Adam_0	policy/DaySurgeon_0	policy/SpC1_0	policy/NightTime_0
policy/UserRoleDeAssignment_1	policy/Mark_1	policy/NightSurgeon_1	policy/SpC1_1	policy/DayTime_0

Fig. 11. Relevant Alloy output after running predicate *UserRoleAssignPred* for the EHR example in Section VII

Next, we activate user Adam in roles DaySurgeon at time DayTime and at location SpC1. At the same time we also deactivate user Adam (if he is active) from role NightSurgeon. This activation and deactivation corresponds to row 9 of Table IV. We introduce the following facts to the alloy model created thus far,

```
fact AdamActivatesDaySurgeonAtDayTimeAtSpC1{
some uract: UserRoleActivation, urdact:
UserRoleDeActivation, u: Adam, r: DaySurgeon, t: DayTime, l: SpC1 |
(u->r -> l-> t) in uract.URAct_member and
(u->r -> l-> t) not in urdact.URDAct_member
}
```

```
fact MarkDeActivatesNightSurgeonAtDayTimeAtSpC1{
some uract: UserRoleActivation, urdact:
UserRoleDeActivation, u: Adam, r: NightSurgeon, t: DayTime, l: SpC1 |
(u->r -> l-> t) not in uract.URAct_member and
(u->r -> l-> t) in urdact.URDAct_member
}
```

It may be noted that these two facts can also be combined into one fact using the $=>$ operator, however, we feel that it is better to structure all facts independent of each other so that they can be reused later. We test the model by running the following predicate and Alloy presents an example where the above facts are true. in order to converse space we do not show the actual output of the Alloy constraint analyzer.

```
pred UserRoleActivationPred (ura, ura': 
UserRoleActivation, urda, urda': UserRoleDeActivation,
u: User, r: Role, l: Location, t: Time){
ura'.URAct_member=ura.URAct_member+ (u->r->l->t) &&
urda'.URDAct_member=urda.URDAct_member- (u->r-> l -> t)
}
run UserRoleActivationPred for 4
```

In order to illustrate the composition of spatial SoD we define spSoD for roles DaySurgeon and NightSurgeon which are conflicting for user Adam. Note that this situation is not defined in the example in Section VII. We introduce the following fact to the above created policy.

```
fact AdamSoDForDaySurgeonAndNightSurgeon {
some r1: DaySurgeon,  r2:NightSurgeon,
u: Adam, uract: UserRoleActivation, l:SpC1, t: DayTime|
(u->r1-> l-> t) in (uract.URAct_member) =>
(u->r2-> l-> t) not in (uract.URAct_member)
}
```

Note that the above fact allows Adam to activate role DaySurgeon but deactivates Adam in role NightSuregeon. Again we execute the predicate *UserRoleActivationPred* and observe the output generated by Alloy. The desired tuples appear in the *UserRoleActivation* signature establishing the enforcement of spatial SoD.

In this section we have demonstrated the proposed methodology for composition of GST-

RBAC policy using the GSTRBAC specification model developed in Section VIII-B. Note that this composition methodology can be employed to analyze policy components before actually implementing in an organization. Also note that an Alloy policy model can help the administrator to add new constraints, permissions, roles to the policy, through out its life time in a consistent and conflict free manner.

IX. Related Work

We discuss the work related to this paper according to two broad categories; firstly, we discuss the state of the art in spatial access control models; secondly, we touch briefly on some of the defining work in the area of specification modeling of software in general and access control models in particular.

A number of researchers have addressed the issues of spatially aware access control mechanisms. One of the pioneering thrusts in this direction was [29], which primarily deals with the application of access control to satellite image maps. More recently, the same authors have proposed an authorization model for geospatial data in [30]. In this work, access to spatial imagery is restricted by defining credentials of subjects and authorization privileges. While this work provides insight into effective management of spatial images by considering both spatial and temporal dimensions, it does not lay down a formal framework for definition of a complete access control policy.

Similarly, an access control model for spatial data on the web is proposed in [31]. The model defines authorization spaces, and grants access to requests for objects only within the authorization space. While this model is effective in controlling access to spatial data on the Web, it lacks flexibility to define relationships (hierarchical or flat) between locations and authorization decisions based on these relationships.

Another effort in the direction of a spatially aware access control model is the definition of environmental roles in [32]. Environmental roles not only define the spatial properties of a role but also defines other contextual properties such as temporal. While environmental roles can be integrated into an RBAC policy, our current methodology of defining spatial constraints and attaching them to existing roles affords simpler extension of existing systems such as GTRBAC.

Closest to the current work is GEORBAC [33], an extension of the RBAC model which defines spatial *roles*, which can be assumed within a defined *spatial extend* or boundary. This

approach is close to the above mentioned environmental roles in [32]. Spatial information with regard to a role is represented as a role schema. However, GEO-RBAC defines one spatial extent for each role, which implies that each spatial location in an organization needs to have its own role, clearly a scalability issue. On the other hand, our proposed GST-RBAC model allows definition of semantic relationship between locations in the form of rich spatial constraints. This approach results in de-linking of the number of roles from the number of locations in an organization, clearly a desirable feature. Further, in our current work we represent spatial information as spatial constraint which can be attached to any role already existing in an access control policy, thus allowing a simpler spatial extension of GTRBAC. GEORBAC also defines hierarchies of roles based on containment of locations. In our work we also define hierarchies of roles but with added ability of defining spatial hierarchy on role hierarchies. *Users* in GEO-RBAC have a position which can be both real and logical. Real position in GEORBAC model is defined as the coordinate position of the user with respect to any earth bound coordinate system. On the other hand, logical position in this model refers to position representation which is independent from the underlying positioning technology and can be computed using position mapping functions. By contrast, in our model virtual location refers to the location defined by membership of a computing group such as an IP pool, with no reference to physical location. Further, we recognize the difference in geometric and symbolic locations and present an access control model which works for these as well as virtual locations. Our current work is also close to [34], in which constraints on role activation are specified using an XML based grammar. In this work constraints are defined as temporal and non-temporal, where spatial constraints are defined as the later. However, [34] does not exploit the relationship between locations and roles based on spatial relations.

Requirements and specification modeling of software has been researched for some time and an excellent overview of classical work and future directions has been presented in [35]. The challenges of software requirement modeling are effectively addressed by [36]–[41], using state transition diagrams, context diagrams, formal languages, and visual representations. A philosophical treatment of the subject can be found in [42]. Some of the work more relevant to this paper is briefly outline next. The use of state transition diagrams for modeling user interaction was first researched by [43] and [44] introduced a modeling technique for capturing entities and events. The interaction of software with the environment was first explored by [45] using Context

Diagrams. A formal framework for modeling agents and their interfaces was proposed in the form of GIST [36], which provided reasoning about individual choice of behavior and responsibility for constraints. The appropriateness of formal specification languages for the description of user interface phenomena has been investigated in [46] and [47]. In [46] a hybrid model and notation to address status and event phenomena symmetrically is proposed. A method for automatically analyzing formal, state-based requirements specifications for some aspects of completeness and consistency has been proposed by [48]. The approach uses a low-level functional formalism, simplifying the analysis process. A taxonomy and some preliminary principles for designing visual representations of formal specifications has been addressed in [49]. They exemplify with the help of an aircraft control system and formulate the question based decision diagrams for representing user requirements.

Recently, the use of policies in the context of network and distributed applications and services has been an area of active research and is considered as an emerging software domain [5]–[9]. Policies have been applied to access control mechanisms [10], to Policy Based Network Management Systems (PBNM) [6], [11], to express the enterprise viewpoint [12] of ODP (Open Distributed Processing), to agent based systems [13], and to Web Services Policy (WS-Policy) [14]. Also, a number of researchers have applied principles of software engineering to policy composition by considering policy as a software artifact [34] [50] [51] [52], [53] [54] but have not addressed the issue related to requirement modeling methodology needed by the end user for context-aware (both temporal and spatial) policies. Further, analogous to generic software applications, systems employing policies also operate in a physical environment and adapt to changing system and environment contextual parameters such as time and location. The effect of environmental constraints on the computing environment has been emphasized by many researchers [20], [41], [55], [56] and has found new impetus in the ubiquitous and pervasive computing paradigms. An analysis of the condition functions or predefined attributes in an access control policy has been presented by [57]. They present the design of the novel Antigone Condition Framework (ACF). This framework implements a general-purpose condition specification, implementation, and evaluation service.

UML [58] has proved quite useful for representing system requirements and models. Some of the notable work in using UML as a modeling environment for secure software has been reported in [59], [60] which propose extensions to UML by defining stereotypes to evaluate

diagrams and to indicate possible vulnerabilities. In [51], researchers have proposed using UML to support role engineering. In [52] a use case diagram is used for representing static view of roles in policies, a useecase diagram for the functional view and a collaboration diagram for the dynamic model. In [54] UML template class diagram is used to capture the structure of policies. An insightful comparison between policies and requirements can be found in [5], along with guiding principles for composing policies with focus on software requirements. Currently, the extension mechanism of stereotypes [58] in UML is employed to represent context parameters other than time. While this approach is satisfactory for a small set of context parameter values, it does not scale well to larger set of values of the same context parameter. For example, privacy policy of electronic health record may dictate separate definition of accesses for a large number of locations in a hospital. Using stereotypes results in a number of similar use case diagrams with different stereotypes. This problem is further exacerbated with endusers defining multiple contextual constraints on a specific policy requirement. Within UML, Object Constraint Language (OCL) is used to specify preeonditions, posteonditions, invariants, and other kind of constraints [61]. While OCL can be used to define a small number of constraints, it is not scalable for multiple dimensions of contextual constraints. In addition, a number of shortcomings of OCL have also been reported [62] which mostly have concern with its expressiveness. While the above mentioned work on composition of policies as software artifacts using UML provides valuable insight into the issues surrounding policy engineering, UML is intrinsically informal and does not provide inbuilt formalism for conflict resolution and hence is deemed unfit to satisfy our stated goals.

Another candidate for access control policy specification modeling is Z modeling language [63]. In Z composition of system properties is done using schema calculus. However Z lacks fully automatic analysis of the composed model in the style of a model checker. In our current work we use Alloy [15] for modeling and conflict resolution of GSTRBAC policies. An exhaustive comparison regarding the strengths and weaknesses of UML, Z and Alloy is provided in [15]. An Alloy model is declarative: it can describe the effect of a behavior without giving its mechanism. It also allows partial models to be created and tested for predefined conflicts. Modeling of access control policies using Alloy has been attempted before [64], however this model is for RBAC [2] without temporal and spatial constraints.

X. Conclusion

In this paper, we have presented GSTRBAC, the spatial extension of GTRBAC. We have formally developed the notion of rich spatial constraints in which participating locations have semantic relationship with each other and access control decisions are dependent on these relationships. We also define spatial separation of duty and role hierarchy with rich spatial constraints. We further analyze the proposed GSTRBAC model by defining conflicts which may arise while using spatial constraints in addition to temporal ones. We exemplify GSTRBAC with an example from the health care domain where location of a user has direct bearing on the access control rights available to him.

In order to analyze the proposed GSTRBAC model, we develop its formal specifications using the light-weight formal modeling environment, Alloy. The specification model is analyzed utilizing the accompanying Alloy constraint analyzer for pin-pointing conflicts and subsequent resolution. Next, we illustrate the composition of an organization's access control policy using the GSTRBAC policy specification model. Conflicts which may arise while composing an access control policy become evident using the Alloy constraint analyzer. We show that simulating an access control policy in a light-weight formal environment, before it is implemented in an organization, helps to uncover unwanted and dangerous flaws. We also demonstrate that conflict resolution for access control policies may be done piece-meal allowing the policy to be analyzed step by step during its engineering phase.

While Alloy offers an excellent formalism for modeling access control specifications, it suffers from performance issues, especially when the policy model includes hundreds of roles. Although, modeling portions of the policy at time mitigates this drawback, a full analysis may become time consuming. Further, the formalism afforded by Alloy is expressive enough for developing an access control model, but it is more desirable to do so using some form of visual tools. This observation is more pertinent since policy is to be composed by a policy administrator who may not have the desire or the inclination to fully understand Alloy syntax. In the future we plan to develop such a visual interface for access control policy specification so that Alloy coupled with this interface can find more applications in the industry.

APPENDIX I
ALLOY SPECIFICATION MODEL OF GENERALIZED SPATIO-TEMPORAL ROLE BASED ACCESS CONTROL

Listing of the Alloy specification of the generalized spatio-temporal role based access control policy

```
module policy
-------------- Signature Declarations --------
sig User {}
sig Role{
}
sig Permission{}
sig Location {operator: SpatialOperators -> Location}
sig SpatialOperators{}
sig Time {}

sig RoleEnable {re_member : Role-> Location -> Time}
sig RoleDisable {rd_member : Role-> Location -> Time}
sig UserRoleAssignment
{URA_member : User -> Role -> Location -> Time}
sig UserRoleDeAssignment
{URDA_member : User -> Role -> Location -> Time}
sig RolePermissionAssignment
{RPA_member : Role-> Permission  -> Location -> Time}
sig RolePermissionDeAssignment
{RPDA_member : Role-> Permission -> Location -> Time}
sig UserRoleActivation
{URAct_member : User-> Role-> Location-> Time}
sig UserRoleDeActivation
{URDAct_member : User->Role->Location->Time }

----------------Facts------------------------
fact UsersEnableNotDisable{ //conflict type 1a
some rd: RoleDisable, re: RoleEnable,
 r: Role, t: Time, l: Location |
((r -> l-> t) in re.re_member =>
(r -> l-> t) not in rd.rd_member &&
(r -> l-> t) not in re.re_member =>
(r -> l-> t) in rd.rd_member)
```

```
}

fact UsersAssignedNotDeassigned{ //conflict type 1b
some urda: UserRoleDeAssignment, ura: UserRoleAssignment,
u: User, r: Role, t: Time, l: Location |
((u->r -> l-> t) in ura.URA_member =>
(u->r -> l-> t) not in urda.URDA_member &&
(u->r -> l-> t) not in ura.URA_member =>
(u->r -> l-> t) in urda.URDA_member)
}

fact RolePermissionAssignedAndDeassigned {   //conflict type 1c
some rpa: RolePermissionAssignment, rpda: RolePermissionDeAssignment,
 p: Permission, r: Role, l:Location, t:Time|
((r -> p-> l-> t) in rpa.RPA_member =>
(r ->p-> l-> t) not in rpda.RPDA_member &&
(r -> p-> l-> t) not in rpa.RPA_member =>
(r -> p-> l-> t) in rpda.RPDA_member)
}

fact RoleEnabledThenAssigned{                    //conflict type 2
some ura: UserRoleAssignment, re:  RoleEnable,
 u: User, r: Role, t: Time, l: Location |
r->l->t in (re.re_member) =>
u -> r->l->t in (ura.URA_member)
}

fact RoleAssignedThenActivated{
some uract: UserRoleActivation, ura: UserRoleAssignment,
u: User, r: Role, t: Time, l: Location |
u -> r->l->t in (ura.URA_member) =>
u -> r->l->t in (uract.URAct_member)
}

fact UsersActivatedNotDeActivated{ //conflict type 1d
some ura: UserRoleActivation, urda: UserRoleDeActivation,
u: User, r: Role, t: Time, l: Location |
((u->r -> l-> t) in ura.URAct_member =>
(u->r -> l-> t) not in urda.URDAct_member &&
(u->r -> l-> t) not in ura.URAct_member =>
```

```
(u->r -> l-> t) in urda.URDAct_member)
}
----------------------Predicates/Functions------------
pred EnableRole (rd, rd': RoleDisable, re, re': RoleEnable,
 r: Role, t: Time, l: Location) {
re'.re_member=re.re_member+ (r-> l -> t) &&
rd'.rd_member=rd.rd_member- (r-> l -> t)
}

pred DisableRole (rd, rd': RoleDisable, re, re': RoleEnable,
 r: Role, t: Time, l: Location) {
re'.re_member=re.re_member- (r-> l -> t) &&
rd'.rd_member=rd.rd_member+ (r-> l -> t)
}
run EnableRole for 3 but 2 RoleEnable, 2 RoleDisable

pred UserRoleAssignPred (ura, ura': UserRoleAssignment,
 urda, urda' : UserRoleDeAssignment, u: User, r: Role,
 t: Time, l: Location, re':  RoleEnable) {
ura'.URA_member=ura.URA_member+ (u->r->l->t)&&
urda'.URDA_member=urda.URDA_member- (u->r-> l -> t)
}

pred UserRoleDeAssignPred (ura, ura': UserRoleAssignment,
urda, urda' : UserRoleDeAssignment, u: User,
r: Role, t: Time, l: Location) {
ura'.URA_member=ura.URA_member- (u->r->l->t) &&
urda'.URDA_member=urda.URDA_member+ (u->r-> l -> t)
}
run UserRoleAssignPred for 2

pred RolePermissionAssignPred (rpa, rpa':
RolePermissionAssignment,
rpda, rpda' : RolePermissionDeAssignment,
r: Role, p:Permission, t: Time, l: Location) {
rpa'.RPA_member=rpa.RPA_member+ (r->p-> l-> t) &&
rpda'.RPDA_member=rpda.RPDA_member- (r-> p-> l-> t)
}

pred RolePermissionDeAssignPred (rpa, rpa':
```

```
RolePermissionAssignment,
rpda, rpda' : RolePermissionDeAssignment,
r: Role, p:Permission, t: Time, l: Location) {
rpa'.RPA_member=rpa.RPA_member- (r->p-> l-> t) &&
 rpda'.RPDA_member=rpda.RPDA_member+ (r-> p-> l-> t)
}
run RolePermissionAssignPred for 2

pred UserRoleActivationPred (ura, ura':
UserRoleActivation,
 urda, urda': UserRoleDeActivation,
 u: User, r: Role, l: Location, t: Time){
ura'.URAct_member=ura.URAct_member+ (u->r->l->t) &&
 urda'.URDAct_member=urda.URDAct_member- (u->r-> l -> t)
}

pred UserRoleDeActivationPred (ura, ura':
UserRoleActivation,
 urda, urda': UserRoleDeActivation,
 u: User, r: Role, l: Location, t: Time){
ura'.URAct_member=ura.URAct_member- (u->r->l->t) &&
 urda'.URDAct_member=urda.URDAct_member+ (u->r-> l -> t)
}


run UserRoleActivationPred for 2

pred SoD (r1, r2:Role, u:User,
uract: UserRoleActivation,
 l:Location, t: Time){
(u->r1-> l-> t) in (uract.URAct_member) =>
(u->r2-> l-> t) not in (uract.URAct_member)
}
run SoD for 2

pred ACPolicy (Adam : User, DaySurgeon: Role, SpC1: Location,
DayTime: Time, re, re': RoleEnable,
rd, rd': RoleDisable,ura, ura': UserRoleAssignment,
 urda, urda' : UserRoleDeAssignment,
rpa, rpa': RolePermissionAssignment,
```

```
rpda, rpda' : RolePermissionDeAssignment, p:Permission,
uract, uract': UserRoleActivation,
urdact, urdact': UserRoleDeActivation ) {
EnableRole (rd, rd', re, re',
DaySurgeon, DayTime, SpC1) and
RolePermissionAssignPred (rpa, rpa', rpda, rpda',
DaySurgeon, p, DayTime, SpC1) and
UserRoleAssignPred (ura, ura', urda, urda', Adam,
DaySurgeon, DayTime, SpC1, re') and
UserRoleActivationPred (uract, uract', urdact,
urdact', Adam, DaySurgeon, SpC1, DayTime)

}
run ACPolicy for 2
------------------------Assertion----------------------
assert CheckRoleEnable {
all rd, rd', rd'': RoleDisable, re, re', re'': RoleEnable, r
: Role, t: Time, l: Location |
no r.(re.re_member) and EnableRole (rd, rd', re, re', r, t, l)
 and DisableRole (rd', rd'', re', re'', r, t, l)
implies re.re_member=re''.re_member
}
check CheckRoleEnable for 2

assert CheckUserRoleAssignment {
all ura, ura', ura'': UserRoleAssignment, urda,
urda', urda'': UserRoleDeAssignment, u: User, r: Role,
 t: Time, l: Location, re: RoleEnable |
no u.(ura.URA_member) and
UserRoleAssignPred (ura, ura', urda, urda', u, r, t, l, re) and
UserRoleDeAssignPred (ura', ura'', urda', urda'', u, r, t, l)
implies ura.URA_member=ura''.URA_member
}
check CheckUserRoleAssignment for 2

assert CheckRolePermissionAssigned {
all rpa, rpa', rpa'': RolePermissionAssignment,
 rpda, rpda', rpda'': RolePermissionDeAssignment,
 r: Role, p: Permission, t: Time, l: Location |
no r.(rpa.RPA_member) and
```

```
RolePermissionAssignPred (rpa, rpa', rpda, rpda', r, p, t, l) and
RolePermissionDeAssignPred (rpa', rpa'', rpda', rpda'', r, p, t, l)
implies rpa.RPA_member=rpa''.RPA_member
}
check CheckRolePermissionAssigned for 2

assert CheckUserRoleActivation {
all uract, uract', uract'': UserRoleActivation, urdact,
urdact', urdact'': UserRoleDeActivation, u: User,
r: Role, t: Time, l: Location |
no (r->t->l->u).(uract.URAct_member) and
UserRoleActivationPred (uract, uract', urdact, urdact', u, r,l, t) and
UserRoleDeActivationPred (uract', uract'', urdact', urdact'', u, r, l, t)
implies uract.URAct_member=uract''.URAct_member
}
check CheckUserRoleActivation for 2

assert ACPolicyAssert  {
lone Adam : User, DaySurgeon: Role, SpC1: Location, DayTime: Time,
re, re': RoleEnable,rd, rd': RoleDisable,
ura, ura': UserRoleAssignment,
urda, urda' : UserRoleDeAssignment,
rpa, rpa': RolePermissionAssignment,
rpda, rpda' : RolePermissionDeAssignment, p:Permission,
uract, uract': UserRoleActivation,
urdact, urdact': UserRoleDeActivation |
EnableRole (rd, rd', re, re', DaySurgeon,
DayTime, SpC1) and
RolePermissionAssignPred (rpa, rpa', rpda, rpda',
DaySurgeon, p, DayTime, SpC1) and
UserRoleAssignPred (ura, ura', urda, urda', Adam,
DaySurgeon, DayTime, SpC1, re') and
UserRoleActivationPred (uract, uract', urdact, urdact',
Adam, DaySurgeon, SpC1, DayTime)
}
check ACPolicyAssert for 2
```

APPENDIX II

ALLOY MODEL OF ROLE ENABLING FOR GSTRBAC BASED ACCESS CONTROL POLICY DEPICTED IN SECTION VII

```
module policy
------------- Signature Declarations --------------
sig User {}
sig Role{}
sig Permission{}
sig Location {
operator: SpatialOperators -> Location}
sig SpatialOperators{}
sig Time {}
sig RoleEnable {
re_member : Role-> Location -> Time}
sig RoleDisable {
rd_member : Role-> Location -> Time}
sig UserRoleAssignment {
URA_member : User -> Role -> Location -> Time}
sig UserRoleDeAssignment {
URDA_member : User -> Role -> Location -> Time}
sig RolePermissionAssignment {
RPA_member : Role-> Permission  -> Location -> Time}
sig RolePermissionDeAssignment {
RPDA_member : Role-> Permission -> Location -> Time}
sig UserRoleActivation {
URAct_member : User-> Role-> Location-> Time}
sig UserRoleDeActivation {
URDAct_member : User->Role->Location->Time }
sig RoleHierarchy {
rh_member: Role -> Role -> Location -> Time}

sig DaySurgeon extends Role{}
sig NightSurgeon extends Role{}
sig DayTime extends Time{}
sig NightTime extends Time{}
sig SpC1 extends Location{}

----------------------Facts--------
fact UsersEnableNotDisable{ //conflict type 1a
```

```
some rd: RoleDisable, re: RoleEnable,
r: Role, t: Time, l: Location |
((r -> l-> t) in re.re_member =>
(r -> l-> t) not in rd.rd_member &&
(r -> l-> t) not in re.re_member =>
(r -> l-> t) in rd.rd_member)
}
-----------------Predicates/Functions---------
pred EnableRole (rd, rd': RoleDisable,
re, re': RoleEnable, r: Role,
t: Time, l: Location) {
re'.re_member=re.re_member+ (r-> l -> t) &&
rd'.rd_member=rd.rd_member- (r-> l -> t)
}
pred DisableRole (rd, rd': RoleDisable,
re, re': RoleEnable, r: Role, t: Time, l: Location) {
re'.re_member=re.re_member- (r-> l -> t) &&
rd'.rd_member=rd.rd_member+ (r-> l -> t)
}
run EnableRole for 3 but 2 RoleEnable, 2 RoleDisable
-----------------Access Control Policy facts -------
fact DaySurgeonEnableAtDayTime{
some rd: RoleDisable, re: RoleEnable,
r: DaySurgeon, t: DayTime, l: SpC1 |
(r -> l-> t) in re.re_member
}
fact DaySurgeonDisableAtNightTime{
some rd: RoleDisable, re: RoleEnable,
r: DaySurgeon, t: NightTime, l: SpC1 |
(r -> l-> t) in rd.rd_member
}
fact NightSurgeonEnableAtNightTime{
some rd: RoleDisable, re: RoleEnable,
r: NightSurgeon , t: NightTime, l: SpC1 |
(r -> l-> t) in re.re_member
}
fact NightSurgeonDisableAtDayTime{
some rd: RoleDisable, re: RoleEnable,
r: NightSurgeon, t: DayTime, l: SpC1 |
(r -> l-> t) in rd.rd_member
```

```
}
```

APPENDIX III

ALLOY MODEL OF USER ROLE ASSIGNMENT FOR GSTRBAC BASED ACCESS CONTROL POLICY DEPICTED IN SECTION VII

The following is the additonal Alloy code added to Appendix II for user role assignment.

```
sig Adam extends User{}
sig Mark extends User{}
-----------------Facts--------------
pred UserRoleAssignPred (ura, ura':
UserRoleAssignment, urda, urda' :
UserRoleDeAssignment, u: User, r: Role,
t: Time, l: Location, re':  RoleEnable)
{
ura'.URA_member=ura.URA_member+ (u->r->l->t) &&
urda'.URDA_member=urda.URDA_member- (u->r-> l -> t)
}

pred UserRoleDeAssignPred (ura,
ura': UserRoleAssignment, urda,
urda' : UserRoleDeAssignment,
u: User, r: Role, t: Time, l: Location)
{
ura'.URA_member=ura.URA_member- (u->r->l->t) &&
urda'.URDA_member=urda.URDA_member+ (u->r-> l -> t)
}
run UserRoleAssignPred for 3

------------Access Control Polciy facts --------

fact AdamAssignedToDaySurgeonAtDayTimeAtSpC1{
some ura: UserRoleAssignment,
urda: UserRoleDeAssignment, u: Adam,
r: DaySurgeon, t: DayTime, l: SpC1 |
(u->r -> l-> t) in ura.URA_member and
(u->r -> l-> t) not in urda.URDA_member
}

fact AdamNotAssignedToDaySurgeonAtNightTimeAtSpC1{
```

```
some ura: UserRoleAssignment,
urda: UserRoleDeAssignment, u: Adam,
r: DaySurgeon, t: NightTime, l: SpC1 |
(u->r -> l-> t) not in ura.URA_member and
(u->r -> l-> t) in urda.URDA_member
}

fact MarkAssignedToNightSurgeonAtNightTimeAtSpC1{
some ura: UserRoleAssignment,
urda: UserRoleDeAssignment, u: Mark,
r: NightSurgeon, t: NightTime, l: SpC1 |
(u->r -> l-> t) in ura.URA_member and
(u->r -> l-> t) not in urda.URDA_member
}

fact MarkNotAssignedToNightSurgeonAtDayTimeAtSpC1{
some ura: UserRoleAssignment,
urda: UserRoleDeAssignment, u: Mark,
r: NightSurgeon, t: DayTime, l: SpC1 |
(u->r -> l-> t) not in ura.URA_member and
(u->r -> l-> t) in urda.URDA_member
}
```

REFERENCES

[1] C. A. Patterson, R. R. Muntz, and C. M. Pancake, "IEEE pervasive computing: Spotlight -challenges in locationaware computing." *IEEE Distributed Systems Online*, vol. 4, no. 10, 2003.

[2] R. S. Sandhu, D. F. Ferraiolo, and D. R. Kuhn, "The NIST model for rolebased access control: towards a unified standard." in *ACM Workshop on Role-Based Access Control*, 2000, pp. 47–63.

[3] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "A Generalized Temporal RoleBased Access Control Model." *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 1, pp. 4–23, 2005.

[4] E. Lupu and M. Sloman, "Conflicts in policybased distributed systems management." *IEEE Trans. Software Eng.*, vol. 25, no. 6, pp. 852–869, 1999.

[5] A. I. Antón, J. B. Earp, T. A. Alspaugh, and C. Potts, "The role of policy and stakeholder privacy values in requirements engineering." in *RE*. IEEE Computer Society, 2001, pp. 138–145.

[6] R. Chadha, G. Lapiotis, and S. Wright, "Policybased networking." *IEEE Network.*, vol. 16, no. 2, pp. 8–9, 2002.

[7] M. Sloman and E. Lupu, "Security and management policy specification." *IEEE Network.*, vol. 16, no. 2, pp. 10–19, 2002.

[8] P. T. Devanbu and S. G. Stubblebine, "Software engineering for security: a roadmap." in *ICSE - Future of SE Track*, 2000, pp. 227–239.

[9] S. Reiff-Marganiec and K. J. Turner, "Feature interaction in policies." *Computer Networks*, vol. 45, no. 5, pp. 569–584, 2004.

[10] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models." *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[11] D. C. Verma, "Simplifying network administration using policy-based management." *IEEE Network*, vol. 16, no. 2, pp. 20–26, 2002.

[12] E. A. Boiten, H. Bowman, J. Derrick, P. F. Linington, and M. Steen, "Viewpoint consistency in ODP." *Computer Networks*, vol. 34, no. 3, pp. 503–537, 2000.

[13] M. Barbuceanu, T. Gray, and S. Mankovski, "Providing telecommunication services through multi-agent negotiation." in *IATA*, 1999, pp. 124–136.

[14] N. Mukhi and P. Plebani, "Supporting policy-driven behaviors in web services: experiences and issues." in *ICSOC*, 2004, pp. 322–328.

[15] D. Jackson, "Alloy: a lightweight object modelling notation." *ACM Trans. Softw. Eng. Methodol.*, vol. 11, no. 2, pp. 256–290, 2002.

[16] MIT, "Alloy case studies," 2003, pp. http://alloy.mit.edu/case–studies.php.

[17] G. D. Abowd and A. J. Dix, "Integrating status and event phenomena in formal specifications of interactive systems." in *SIGSOFT FSE*, 1994, pp. 44–52.

[18] U. Leonhardt and J. Magee, "Multi-sensor location tracking." in *MOBICOM*, 1998, pp. 203–214.

[19] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system." *ACM Trans. Inf. Syst.*, vol. 10, no. 1, pp. 91–102, 1992.

[20] B. N. Schilit, N. Adams, R. Gold, M. M. Tso, and R. Want, "The PARCTAB mobile computing system." in *Workshop on Workstation Operating Systems*, 1993, pp. 34–39.

[21] G. D. Abowd, C. G. Atkeson, J. I. Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: A mobile context-aware tour guide." *Wireless Networks*, vol. 3, no. 5, pp. 421–433, 1997.

[22] M. J. Egenhofer and R. D. Franzosa, "Point set topological relations." *International Journal of Geographical Information Systems*, vol. 5, pp. 161–174, 1991.

[23] U. Leonhardt, J. Magee, and P. Dias, "Location service in mobile computing environments." *Computers & Graphics*, vol. 20, no. 5, pp. 627–632, 1996.

[24] G. Dommety and R. Jain, "Potential networking applications of global positioning systems (GPS)," *CoRR*, vol. cs.NI/9809079, 1998.

[25] D. Jackson, "Alloy: A logical modelling language." in *ZB*, 2003, p. 1.

[26] D. Jackson, I. Shlyakhter, and M. Sridharan, "A micromodularity mechanism." in *ESEC / SIGSOFT FSE*, 2001, pp. 62–73.

[27] S. W. Xu, "Modeling the active badge system with alloy and its automatic constraint analyzer," 2003, p. http://alloy.mit.edu/contributions/ActiveBadgeInAlloy.pdf.

[28] MIT, "Alloy faq," 2003, p. http://alloy.mit.edu/faq.php.

[29] S. A. Chun and V. Atluri, "Protecting privacy from continuous high-resolution satellite surveillance." in *DBSec*, 2000, pp. 233–244.

[30] V. Atluri and S. A. Chun, "An authorization model for geospatial data." *IEEE Trans. Dependable Sec. Comput.*, vol. 1, no. 4, pp. 238–254, 2004.

[31] E. Bertino, M. L. Damiani, and D. Momini, "An access control system for a web map management service." in *RIDE*, 2004, pp. 33–39.

[32] M. J. Covington, W. Long, S. Srinivasan, A. K. Dey, M. Ahamad, and G. D. Abowd, "Securing context-aware applications using environment roles." in *SACMAT*, 2001, pp. 10–20.

[33] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca, "GEO-RBAC: a spatially aware RBAC." in *SACMAT*, 2005, pp. 29–37.

[34] R. Bhatti, E. Bertino, and A. Ghafoor, "X-FEDERATE: A policy engineering framework for federated access management." *IEEE Trans. Software Eng.*, vol. 32, no. 5, pp. 330–346, 2006.

[35] B. Nuseibeh and S. M. Easterbrook, "Requirements engineering: a roadmap." in *ICSE - Future of SE Track*, 2000, pp. 35–46.

[36] M. S. Feather, "Language support for the specification and development of composite systems." *ACM Trans. Program. Lang. Syst.*, vol. 9, no. 2, pp. 198–234, 1987.

[37] K. Yue, "What does it mean to say that a specification is complete?" in *IWSSD-4*, 1987.

[38] W. N. Robinson, "Integrating multiple specifications using domain goals." in *IWSSD-5*, 1989, pp. 219–225.

[39] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition." *Sci. Comput. Program.*, vol. 20, no. 1-2, pp. 3–50, 1993.

[40] M. Jackson, "The meaning of requirements." *Ann. Software Eng.*, vol. 3, pp. 5–21, 1997.

[41] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications." in *1st Intl. Workshop on Mobile Computing Systems and Applications*, 1994, pp. 85–90.

[42] P. Zave and M. Jackson, "Four dark corners of requirements engineering." *ACM Trans. Softw. Eng. Methodol.*, vol. 6, no. 1, pp. 1–30, 1997.

[43] S. Wasserman, "A specification method for interactive information systems." *IEEE Catalog No. 79 CH1401-9C*, pp. 68–79, 1979.

[44] J. A. B. Jr., "Information modeling in the context of system development." in *IFIP Congress*, 1980, pp. 395–411.

[45] P. T. Ward, "The transformation schema: An extension of the data flow diagram to represent control and timing." *IEEE Trans. Software Eng.*, vol. 12, no. 2, pp. 198–210, 1986.

[46] L. J. Bass, G. D. Abowd, and R. Kazman, "Issues in the evaluation of user interface tools." in *ICSE Workshop on SE-HCI*, 1994, pp. 17–27.

[47] S. M. Easterbrook, R. R. Lutz, R. Covington, J. Kelly, Y. Ampo, and D. Hamilton, "Experiences using lightweight formal methods for requirements modeling." *IEEE Trans. Software Eng.*, vol. 24, no. 1, pp. 4–14, 1998.

[48] M. P. E. Heimdahl and N. G. Leveson, "Completeness and consistency in hierarchical state-based requirements." *IEEE Trans. Software Eng.*, vol. 22, no. 6, pp. 363–377, 1996.

[49] N. Dulac, T. Viguier, N. G. Leveson, and M.-A. D. Storey, "On the use of visualization in formal requirements specification." in *RE*, 2002, pp. 71–80.

[50] R. Crook, D. C. Ince, and B. Nuseibeh, "On modelling access policies: Relating roles to their organisational context." in *RE*, 2005, pp. 157–166.

[51] P. Epstein and R. S. Sandhu, "Towards a UML based approach to role engineering." in *ACM Workshop on Role-Based Access Control*, 1999, pp. 135–143.

[52] M. E. Shin and G.-J. Ahn, "UML-based representation of role-based access control." in *WETICE*, 2000, pp. 195–200.

[53] *5th IEEE International Symposium on Requirements Engineering (RE 2001), 27-31 August 2001, Toronto, Canada*. IEEE Computer Society, 2001.

[54] D.-K. Kim, I. Ray, R. B. France, and N. Li, "Modeling role-based access control using parameterized UML models." in *FASE*, 2004, pp. 180–193.

[55] G. J. F. Jones and P. J. Brown, "Information access for context-aware appliances." in *SIGIR*, 2000, pp. 382–384.

[56] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness." in *HUC*, 1999, pp. 304–307.

[57] P. D. McDaniel, "On context in authorization policy." in *SACMAT*, 2003, pp. 80–89.

[58] I. Jacobson, G. Booch, and J. E. Rumbaugh, "Excerpt from "the unified software development process": The unified process." *IEEE Software*, vol. 16, no. 3, pp. 82–90, 1999.

[59] J. Jürjens and G. Wimmel, "Security modelling for electronic commerce: The common electronic purse specifications." in *I3E*, 2001, pp. 489–506.

[60] J. Jürjens, "Towards development of secure systems using UMLsec." in *FASE*, 2001, pp. 187–200.

[61] A. Kleppe, J. Warmer, and S. Cook, "Informal formality? the object constraint language and its application in the uml metamodel." in *UML*, 1998, pp. 148–161.

[62] M. Vaziri and D. Jackson, "Isome shortcomings of OCL, the object constraint language of UML." in *Response to Object Management Group's request on Information in UML 2.0*, 1999, p. http://people.csail.mit.edu/dnj/publications/omg.pdf.

[63] J. P. Bowen, M. G. Hinchey, and D. Till, Eds., *ZUM '97: The Z Formal Specification Notation, 10th International Conference of Z Users, Reading, UK, April 3-4, 1997, Proceedings*, ser. Lecture Notes in Computer Science, vol. 1212. Springer, 1997.

[64] J. Zao, H. Wee, J. Chu, and D. Jackson, "RBAC schema verification using lightweight formal model and constraint analysis," 2002, pp. http://alloy.mit.edu/case–studies.php.