Multi-layer Human-in-the-loop Service Management Utilizing Rule-based Policies and Set Theoretic Expressions

S. Yousaf Shah†, Christos Parizas*, Geeth De Mel§, Boleslaw Szymanski†, Alun Preece*, Dominic Harries‡, John Ibbotson‡
Stephen Pipes‡
†Rensselaer Polytechnic Institute, Department of Computer Science, Troy, NY, USA, Email: {shahs9, szymab}@rpi.edu
*School of Computer Science and Informatics Cardiff University, UK, Email: {C.Parizas, A.D.Preece}@cs.cardiff.ac.uk
§IBM T.J. Watson Research Center, Yorktown Heights, NY, USA, Email: {grdemel@us.ibm.com}
‡IBM Hursley, UK, Email: {dharries, jbibbotson, pipessd}@uk.ibm.com

***Abstract*— Services management in large-scale service oriented sensor networks carries many challenges. Such networks consist of heterogeneous assets that host a variety of services and maintain perpetual information flow/fusion between hard (i.e. sensing devices, distributed databases) and soft (human) resources. Regulating access to assets and services using policies has proven to be effective in many ways especially when resources are affiliated with multiple parties in coalition formations. In this paper, we propose a two-layer novel service management mechanism: the first layer directly interfaces with humans and is based on controlled natural language (CNL) technologies; and the second interacts with the system level policies and are interpreted utilizing Restriction Set Theoretic Expressions (RSTE). The RSTE layer is responsible for the policy evaluation by limiting the resources available for service compositions pertaining to a user's requests, and facilitates the reporting to the upper-layer of over-restrictive policy constraints that may hinder the service compositions process. Finally, we describe a policy relaxation mechanism between coalition collaborating parties, when services are not configurable given the strict policies currently enforced. The relaxation mechanism exploits the ability of RSTE to flow information from service composition to upper layer, which interacts with human-in-the-loop in a transparent way.**

I. Introduction

Coalitions are formed to achieve common goals, be they humanitarian or combat in nature. The key to success in these goals is the improved situational awareness obtained — in a given context — by fusing available information. Such information is gathered by applying multiple analytics (atomic or composite) over the data gathered from the resources deployed in the field. However, access to these data and analytics are controlled by means of policies. This is due to the inherent need in coalitions to protect individual interests whilst collaborating with others. Typically, there are two types of policies: *high-level policies* governing the interaction among organizations, and *low-level policies* governing the system level constraints, especially in-terms of access control to resources and resulting services. High-level policies are authored by human with some assistance from automated mechanisms to detect violations, conflicts, and so forth. Therefore, we need a human friendly but machine interpretable representation for such policies so that humans could author them whilst being assisted by intelligent mechanisms which can guide them through the process. On the other hand, low-level policies are system level constraints which need be enforced and evaluated in a rapid manner to perform particular tasks, and requires little or no reasoning. This yields to the need of having a representation that can assists in efficient policy enforcement.

In this paper, we propose an initial framework to address both these policy layers: *(a)* a high-level policy specification layer to negotiate and relax policy constraints which hinders the information gathering; and *(b)* a low-level policy specification and enforcement layer which instantiate the required resources to achieve goals. The high-level policies are represented and reasoned with a controlled natural language (CNL) — specifically ITA Controlled English — which makes the communication across organizations transparent and human scrutable whilst the system level policies are represented and enforced utilizing a set theocratic approach called *Restriction Set Theoretic Expressions (RSTE)*. It is important to note that inability to perform a particular request at the lower-level may have resulted due to a higher-lever policy constraint. Thus, it is important to have mechanisms to communicate such issues to the organization level so that appropriate policy negotiations and relaxations can take place in support of the goals. Therefore, the goal of the proposed framework is to provide mechanisms to relax, and transform high-level user-specified policies into low-level controls and vice versa.

In order to illustrate our approach, in this paper, we consider a system for service configuration in a mobile sensor network. Each service hosted in the sensor network produces one or more outputs and require zero or more inputs from other services, thus composite services are hierarchical in nature forming a workflow graph for data processing. In a coalition environment, services are owned by different collaborating organizations or partners, these services are hosted on sensor nodes owned by different partners. Policy makers define policy rules to restrict or allow access to the resources in the network.

*Portions of this paper are taken from: Shah, S. Yousaf, Boleslaw Szymanski. Dynamic Policy Enforcement using Restriction Set Theoretic Expressions (RSTE). IEEE Military Communications (MilCom), 2014, To appear in conference proceedings [1].

Users make requests for services with certain requirements, the requests are handled by the configuration system and based on policies and users' authorization corresponding services are configured for the requestor.

The remainder of the paper is organized as follows. In Section II, we present the related work. Section III presents the high level policy representation. In Section IV, we provide detailed description of *RSTE* and Section V presents the implementation details on how *RSTE* can be implemented using a relational database. Section VI provides an illustrative scenario which describes the management of a service based system, while section VII describes the service management flow between the two layers. Finally, Section VIII concludes the paper and discusses future work.

II. Related Work

Asset sharing and management in coalition environments has been studied in the past and various techniques for resource allocation have be suggested by researchers [2], [3]. Policy enforcement and management techniques for different computing paradigms have been proposed in the past [4], [5], [6]. In [5], [6], more programatic and standardized approaches have been proposed, however, the policy description and representation is closer to computer programs and is therefore not very friendly to non-technical users. Goal oriented and formal logic based frameworks have also been proposed in the past for policy management [7], but their capabilities are limited by the underlaying OWL version. Other semantic representation based frameworks such as [8], [9] have also been proposed. Deontic logic based approaches to address paradoxes in Standard Deontic Logic (SDL) have been proposed in [10], however, our approach differs from deontic logic based approaches and other formal logic based approaches such as [11], [7] in policy representation as well as enforcement because the latter are not feasible for dynamic compositions of services. In [12] attribute based policy enforcement has been investigated, but this approach also utilizes [5] and has a more programming approach for policy representation, also it does not support backtracking and suggestions for policy relaxation.

III. CE Expressed Policies

To implement the presentation layer of the service management mechanism which interfaces with a human-in-the-loop, we utilize a form of CNL known as ITA Controlled English (CE) [13]. Since we have not tested and compared the understandability of CE with the understandability of other lower level well-known policy languages such as Ponder and CIM-SPL through formal experiments, we cannot safely claim that CE is a user friendlier representation than its predecessors. However, there is related experimentation work reported in the literature [14], [15], [16] for testing the friendliness of CNL languages similar to CE to humans and comparing them with formal languages such as OWL. The results of these experiments in all cases led to the conclusion that CNL can do better compared to formal languages especially in situations where users have little or no technical training.

As described in [17] a policy representation using CE complies with the essential requirements of a policy language in that a) it is machine processable b) it is sufficiently expressive to capture policies across different domains, c) it supports decisions utilizing its reasoner and d) it is amenable for analysis for the detection and resolution of conflicting relationships among policies.

In order to explain the multi-layer service management mechanism and how the two policy layers interact with each other, we describe the capabilities of the presentation layer and the basic components of the CE expressed policies. Being a first-order logic representation, CE is capable of defining domain models as a set of concepts, their properties and the relationships between them. It also supports multiple inheritance and can structure hierarchies of concepts. Once the domain model is built, it then can be instantiated by asserting facts.

We use CE for expressing high-level, attribute-based policy rules following the "if - condition - then - action" form. If the conditions are valid, then the embedded CE reasoner defines what activities are required by the system in order to comply with the enforced policies. The IBM Controlled Natural Language Processing Environment[1] or briefly CE Store is a web application which provides an information-processing environment within which human and machine agents (i.e. Java coded entities) can develop and interact with conceptual models described in CE. Within the CE Store, different types of agents can decide what policies must be applied in a given situation. Logical inference rules (i.e. policy rules) can be described and executes using the pre-developed conceptual models.

Policy rules expressed as CE are attributes of the pre-defined domain models. Attributes are sets of concepts, their properties and relationships that are used to describe the system being managed through polices. Each policy rule consists of a quadruple of grammatical blocks:

- Subject: specifies the entities (human/machine) which interpret obligation policies or can access resources in authorization policies (concepts and properties of the conceptual model).
- Action: what must be performed for obligations and what is permitted for authorization (relationships between concepts of the conceptual model)
- Target: objects on which actions are to be performed (concepts and properties of the conceptual model).
- Condition(s): boolean conditions under which the policy is applied (Condition expressions are either CE sentences statemens or expressions of the form $\alpha \lhd \beta$ where α and β are constants, i.e. concepts and properties of the conceptual model or numerical values and $\lhd$ is any of the symbols $>$, $<$, =, $\geqslant$, $\leqslant$, $\neq$)

[1] https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=558d55b6-78b6-43e6-9c14-0792481e4532

In addition, each policy rule contains some descriptive metadata such as the policy Id, the time and date of its creation and its author.

We present a simple authorization policy rule expressed in both CIM-SPL in Table I and CE in Table II to show the different levels of human-friendliness of the two approaches. The policies reflect a simple scenario in a coalition operation context where an authorization policy which allows a user to access a service if and only if the user and the asset are both affiliated with the same partner.

Subject: user
Action: canAccess
Target: service
Condition: users affiliation == service's affiliation

Any non-technical user with little knowledge of the domain model can read and understand the policy rule implemented in CE. The CE representation is not far away from the policy's plain-text explanation provided above. On the contrary, in order for a user to understand the policy rule implemented in CIM-SPL, some technical-programming skills are needed.

IV. The **RSTE** Language

In this section, we present a *Restriction Set Theoretic Expressions (RSTE)* language that is simple yet expressive enough to map high-level policies to lower level system constraints so that lower level execution engines can produce the desired result. This set language is understandable by both higher layers of the framework dealing directly with the user and low level system engines. Therefore, this language enables a two way information flow from a user to the system level objects and back. Figure 1 shows layered view of the three major components of the system.

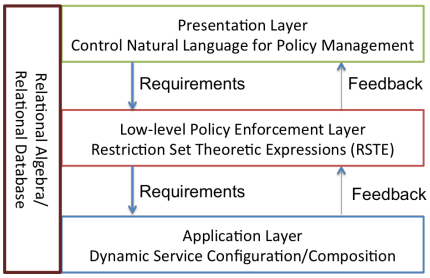

Fig. 1: Set Language is incorporated as layer between Presentation layer and Application

The *Dynamic Service Configuration/Composition* layer is responsible for producing policy enforced, low cost composite services based on user requirements. This layer deals with the input/output matching of different services as well as spatial relevancy constraints to produce composite services that are cost effective and geo-spatially relevant to the user's interests. In cases when this layer fails to produce a service configuration as requested by the user, it notifies the *Controlled Natural Language for Policy Management* via *RSTE* layer about the need for appropriate actions. The *Dynamic Service Configuration/Composition* layer accesses resources that are policy constrained by the *RSTE* layer, therefore this layer considers only resources that are accessible by the requesting user in compliance with policies. If the resources are too scarce for service composition, it generates an output message that is sent to the presentation layer.

A. The Semantics and Operations of ***RSTE*** Language

In this section, we define *RSTE* operations and semantics for *RSTE* sets. A request for service is represented by the $SREQ$ set and its corresponding service response is represented by $SRes$. Following are the sets used by *RSTE* for representing assets, policies, requests/response and users. Alongside these sets we define various set operations, such as *Union, Intersection, Subtraction, Cardinality*, that are performed on these sets to produce a restricted set of assets which is then provided to the system layer for configuration.

Service Set = SS= {{ ServiceName,NodeId, GeoSpatialCoverage,Capabilities, Ownership}}

Node Set = NS = {{ NodeId, Ownership, Location, Conditions, Permissions}}

Service Request Set = SREQ = {{ RequestId, UserId, ServiceName, Capabilities, Properties, Restrictions}}

Policy Set = PS = {{PolicyId, ServiceOwnership, ServiceName, Conditions, UserAffiliation, Restrictions, Action}}

User Set = US ={{UserId, UserName, UserAffiliation, Role, UserProperties}}

RSTE Response Set = RRS= {{ ServiceName, NodeId, GeoSpatialCoverage, ServiceCapabilities, ServiceOwnership, PolicyId, PolicyConditions, UserAffiliation, PolicyRestrictions, RequestId, UserId, SReqCapabilities, SReqRestrictions, Action}}

Service Response Set = SRes = {{ RequestId, Logs, ReturnValue, Failures}}

B. Definitions of the Sets

1) *Service Set (SS)*: This set represents a snapshot of the available services. It is subset of all the services hosted in the asset database. During the set operations this set is filtered based on policies and restrictions applied by the user.
2) *Node Set*: This set represents nodes that host member services of *SS*. It is used to access detailed information about the infrastructure hosting the service and will be used as needed.
3) *Service Request Set*: This set describes service requests made by users.

TABLE I: CIM-SPL representation

Condition
{
subject.affiliation() == target.affiliation()
}
Decision
{
canAccess.allow()
}

TABLE II: CE representation

if
(the service S is affiliated with the partner P) and
(the user U is affiliated with the partner P)
then
(the user U canAccess the service S)
.

TABLE III: Service Set

Set Element Name	Description
Service Name	Name of the service, e.g., SERVICE-1
NodeId	Unique ID of node hosting the service
GeoSpatialCoverage	Geospatial coverage provided by the service
Capabilities	Capabilities provided by the service, e.g, {<Sensor, Video>, <Resolution, HD>}
Ownership	Organization that owns the service, e.g., CW, FredFarm.

TABLE IV: Node Set

Set Element Name	Description
NodeId	Unique ID of node hosting the service
Ownership	Organization that owns the node, e.g., CW, FredFarm.
Location	Physical Location of the Node, e.g., $< LAT, LONG, ALT >$
Conditions	Conditions that applies to usage of the Node.
Permissions	Permissions on this node

TABLE V: Service Request Set

Set Element Name	Description
RequestId	Unique ID of the service requests to distinguish different requests.
UserId	Unique ID of the user who requested the service.
ServiceName	Name of the service that has been request.
Capabilities	Capabilities that are required by the service, e.g., High resolution video
Properties	Configuration properties of the service, e.g., $\{< ModeofOperation, Distributed >\}$
Restrictions	Restrictions on the requested service, e.g., $\{< Ownership, CW >\}$

TABLE VI: Policy Set

Set Element Name	Description
PolicyId	Unique ID of a policy
ServiceOwnership	Owner organization of the service affected by the policy, e.g, FredFarm.
ServiceName	Name of the service affected by the policy.
Conditions	Conditions on the use of the service, e.g., {#service instances < 5}
UserAffiliation	Affiliation of the user requesting the service, e.g., CW or FredFarm
Restrictions	Restrictions on the use of the service, e.g., $\{< Role, FarmOwner >\}$
Action	Access to service based on the policy, i.e., Allow or Deny

4) *Policy Set*: The *Policy Set* represents policies that need to be applied to the *SS* using various *RSTE* operations. It captures various other aspects of the policies (e.g., restrictions) along with *conditions* and *actions*.
5) *User Set*: The *User Set* describes users of the network. Each user has certain properties as well as an associated organization. The *UserId* enables the set to access more detailed information about the user from the detailed assets database in case they are needed. This set is primarily used for policy validation through *RSTE* operations. Various conditions are matched based on the user profile before authorizing a user to access services.
6) *RSTE Response Set*: The *RSTE Response Set* is the set that is supplied to the system layer along with the *Service Request Set (SREQ)* for final service configuration. It contains services that fulfill the policy and user requirements.
7) *Service Response Set*: The *Service Response Set* is primarily for feedback purposes. It is supplied to the *CNL Layer* in order to be presented to the user or to negotiate/relax policies.

C. The RSTE Operations

The *RSTE* operations are like normal *Set Operations* but they can also be performed based on a specific condition (which can be imposed on set members) which is checked in the corresponding sets. Consider two sets A and B, members of these sets are also sets with same semantics. Let {e-1,e-2,e-3} be the semantics of sets A and B.
$A = \{\{1, 2, 3\}, \{a, b, c\}\}$
$B = \{\{1, 5, 10\}, \{e, f, g\}, \{a, x, y\}, \{a, b, c\}\}$

1) **Union** ($\cup$) The union operation combines two sets into one. An union operation is defined as,
$A \cup B = \{x : x \in A \text{ or } x \in B\}$
A union with a condition ν can be written as,
$(A \cup B)_\nu = \{x : x \in A \text{ or } x \in B \text{ and } \nu = true\}$
A simple union of the two sets will be as follows,
$A \cup B = \{\{1, 2, 3\}, \{a, b, c\}, \{a, x, y\}, \{1, 5, 10\}, \{e, f, g\}\}$
A union operation defined on a condition "ν: (A.e-1=B.e-1)", produces following set,
$(A \cup B)_\nu = \{\{1, 2, 3\}, \{1, 5, 10\}, \{a, b, c\}, \{a, x, y\}\}$
2) **Intersection** ($\cap$) The intersection of two sets results into a set with common members among the sets.An intersection operation is defined as,
$A \cap B = \{x : x \in A \text{ and } x \in B\}$
An intersection with a condition ν is written as,
$(A \cap B)_\nu = \{x : x \in A \text{ and } x \in B \text{ and } \nu = true\}$

TABLE VII: User Set

Set Element Name	Description
UserId	Unique ID of a user.
UserName	Name of the user.
UserAffiliation	Affiliated organization of the user, e.g., CW, FredFarm
Role	Role of the user in organization or in a mission, e.g., $< Role, Investigator >$
UserProperties	Other properties of the user, e.g., $< SkillSet, Expert >$

TABLE VIII: RSTE Respnse Set

Set Element Name	Description
Service Name	Name of the service, e.g., SERVICE-1
NodeId	Unique ID of node hosting the service
GeoSpatialCoverage	Geospatial coverage provided by the service
ServiceCapabilities	Capabilities provided by the service, e.g., {<Sensor, Video>, <Resolution, HD>}
ServiceOwnership	Organization that owns the service, e.g., CW, FredFarm.
PolicyId	ID of policy applied to this service.
PolicyConditions	Conditions from policy applied to this service.
UserAffiliation	Affiliation of the user, e.g., CW, FredFarm.
PolicyRestrictions	Restrictions specified by policy.
RequestId	ID of the user's request.
UserId	ID of the user of this service.
SReqCapabilities	Capabilities requested in the Service Request
SReqRestrictions	Restrictions specified in the Service Request.
Action	Authorization action of policy on this service.

A simple union of the two sets is,
$A \cap B = \{\{a, b, c\}\}$
An union operation defined on a condition "ν: (A.e-1=1)", produces an empty set.
$(A \cap B)_\nu = \{\}$

3) **Difference ($-$)** The difference or subtraction is the relative complement of set B in A. The difference operation is defined as,
$A - B = \{x : x \in A \; and \; x \notin B\}$
A difference operation with a condition ν is written as,
$(A - B)_\nu = \{x : x \in A \; and \; x \notin B \; and \; \nu = true\}$
A simple difference of the two sets will be as follows,
$B - A = \{\{1, 5, 10\}, \{e, f, g\}, \{a, x, y\}\}$
A difference operation defined on a condition "ν: (A.e-1=a)", produces set.
$(B - A)_\nu = \{\{a, x, y\}\}$
4) **Cardinality (| |)** Cardinality of a set gives the size of a set. In above example, $|A| = 2$ and $|B| = 4$.

D. Operations for Policy Restrictions

First we need to identify policies that apply to a particular user, then we will be able to find out which services a specific user is authorized to access subject to conditions and restrictions. Moreover, there may be services that are not policy enforced and therefore available to all the users to use. In order to find policies that need to be applied to a service configuration request, we apply series of operations to the policy set (PS).

TABLE IX: Service Response Set

Set Element Name	Description
RequestId	Unique ID of the service request
Logs	Any logs produced by the service configuration at the system level.
ReturnValue	Exit state of the service configuration, e.g, 0,1,-1.
Failures	In case of failed configuration, reason of failure.

1) $PS_1 = (PS \cap US)_\nu \qquad \nu = UserAffiliation$
2) $PS_2 = (PS_1 \cap US)_\nu \; \nu = Role \wedge UserProperties$
3) $SS_{PolicyEnforced} = (SS \cap PS)_\nu$
   $\nu = ServiceName \wedge ServiceOwnership$
4) $SS_{PolicyAllowed} = (PS_2 \cup SS_{PolicyEnforced})_\nu$
   $\nu = ServiceName \wedge Ownership$

Now we find all those services that have no restrictions on them or there is no policy defined to restrict them. These services are available to all the users. To find services with no policy restrictions, we perform following operations.

5) $SS_{NoPolicyEnforced} = (SS - SS_{PolicyEnforced})$
6) $SS_{AllAllowedServices} = (SS_{PolicyAllowed} \cup SS_{NoPolicyEnforced})$
7) $SS_1 = (SS_{AllAllowedServices} \cup SREQ)_\nu$
   $\nu = Capabilities$
8) $RSTE\_SRes = SS_1$ Process based on restrictions and conditions

V. RSTE Implementation

We have implemented the prototype of *RSTE* language using a relational database. The *Relational Algebra* in relational databases provides a natural support for *RSTE* sets and operations, therefore the capabilities of relational databases can be used for an efficient implementation of *RSTE*. The sets in the language and tables in a relational database are analogous. We represent every set by a corresponding table. We then perform different operations on the tables using relational algebra and create views for intermediate steps. These views are then systematically processed to produce other views and finally the view that represents the *RSTE Response Set* which is further processed before providing it to the service configuration layer. For some of the operations shown in section IV-D, we have created "Views" in the database, whereas the operations that deal with a temporal requirement or that need more fine grain processing e.g., Step-6, are done via post-processing on the *RSTE Response Set*.

When a service request is received from the user, it is represented by the *SREQ* set and a corresponding entry is made in the database table. The views automatically get updated and set of all allowed services are filtered out. Now, if the user has defined any specific restrictions or there are policies related to the user requesting a service, further processing is done via code after which allowed services are provided to the *Application Layer* for service configuration. For certain

services or users, very restrictive sharing policies may be defined by the policy makers. Such overly restrictive policies limit the services available for configuration and these services might not be sufficient for a successful configuration leading to failure in meeting the user's request. In such cases, the *RSTE* uses a backtracking mechanism to find out condition(s) that can be relaxed in order to produce a configuration. The condition(s) are then sent to the policy management layer for policy negotiation. When such a condition is relaxed in the policy, the resources are immediately available to the user and services can then be configured to enforce the updated policy.

A. RSTE Backtracking

Policies specified by policy makers might be very restrictive which may constrain resources necessary for service configuration. In a very restrictive case, operations performed in step by step policy application by RSTE may lead to empty sets or empty views in our implementation. Which means that there are no services available for configuration. At any such step where we detect an empty set we back track to previous step and negate the conditions to see if that leads to a non-empty set. If so we can suggest this negation to *CE layer* as relaxation for polices involved. In case there are more than one conditions applied at a particular step, each condition is negated separately and the resultant set is checked for not being empty, this way relaxation can be performed at more granular level and only over-restrictive conditions will be negotiated/relaxed.

Figure 2 shows the backtracking mechanism in the *RSTE* implementation. As shown in the figure 2, when condition "C4" of the policy is applied at "Step4" all services are filtered out yielding an empty set; at this stage the policy enforcement backtracks and applies negotiation of condition "C4" to go to "Step4N". If at this stage the result set is not empty then condition "C4" is restricting the solution and it is propagated to *Presentation Layer* for policy negotiation.

VI. Vignette and Analysis

We provide an illustrative walkthrough using a vignette which describes the management of a service based system that monitors the water quality within a geographic area. The aim of the managed system is to provide intelligence to decision makers for detecting and responding to water contamination incidents at early stages in order to effectively reduce its consequences. We use the steps of the vignette to illustrate the interaction between the high-level, presentation layer policy rules expressed in CE and the system layer policies expressed in RSTE while managing a mixture of heterogenous and distributed open source and hard sensing resources. The involving parties in the vignette are:

- Water Company (CW): responsible for monitoring processing and disseminating water within the geographic area
- Individual owners: water consumers who have their own sensing devices deployed on water distribution for private water monitoring (e.g. farmers)

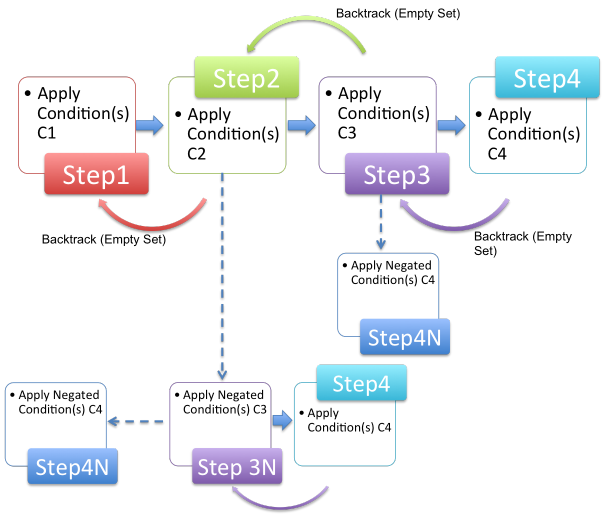

Fig. 2: Backtracking in for policy relaxation in case of overly restricted policies

- Twitter users: water consumers

There are two services deployed by CW for monitoring and responding to water contamination incidents.

Composite Service S1 - Twitter analysis. Is an open source sensing service that performs a real time content analysis of tweets by users tweeting from the geographic area of interest. It is an inexpensive service that performs 24/7.

Composite Service S2 - Smart sensor water monitoring. Is a sensor data collection and analysis service. The smart sensors where S2 partially runs on are owned and operated by CW and are spread throughout the water distribution network. They are chemical sensors (i.e. analytical devices that can provide information about the chemical composition of water) that sample, measure and analyze a dozen of water quality parameters, which are then disseminated via 3G cellular network to the CW control room. The water sensing infrastructure is augmented by a number of additional sensors owned by private individuals (e.g. farmers) who collect data from water sources on their land but do not normally make the collected data available to any party such as the CW.

As mentioned before the behavior of the system is managed through policies. The policies declare what actions the system services are obliged to do and what resources they are allowed or prohibited to access. We present only a small subset of these policies in the timeline steps below.

A. Vignette timeline steps

Step 1: Composite Service S1 performs real time tweets analysis and increases the counter *CNT* every time the word "water" is mentioned in a tweet's body, recording also its geolocation information. Composite Service S2 collects distributed sensing data every 1 hour.

Step 2: Composite Service S1 detects a high number of suspicious tweets, it changes the operation status from normal to critical and triggers Service S2 with different policies which

now require it to be executed every 5 minutes. Composite Service S2 uses only devices owned by CW.

Step 3: The increased sampling rate reveals contamination levels outside the normal range from some of the services in S2. Composite Service S2 attempts to increase the coverage of the area by including services owned by private individuals but owners' privacy policies restrict the services being accessed, thus Composite Service S2 fails.

Policy Rule 1 below is an authorization policy rule, authored by individual owner Fred and expressed in CE which declares his strict access policy as explained above. According to it only the owner of the sensing device (i.e. Fred) can access the water monitoring service that run on it.

Rule 1

```
if
(there is a service named Service1) and
(the user U owns the service Service1)
and
(the user U is affiliated with 'CW-Fred') and
(the user U has the value 'Fred' as name)
then
(the user U can access service Service1).
```

Step 4: The Service Composition component defines which policy/ies is/are restricting composition, and backtracks them by negating conditions.

Step 5: Relaxed RSTE represented policy/ies is/are interpreted in CE and passed to the individuals asking them to approve in order for Composite Service S2 to be deployed.

Policy Rule 1' below is the relaxed version of the restricted Policy Rule 1 after applying the RSTE Backtrack mechanism on it. Rule 1' after negating its fourth condition (the user U has the value 'Fred' as name) allows any user affiliated with 'CW-Fred' (i.e. Water company and individual owner Fred) to access Service1. Composite Service S2 utilizing devices owned by user Fred can now access enough resources in order to be implemented.

Rule 1'

```
if
(there is a service named Service1) and
(the user U owns the service Service1)
and
(the user U is affiliated with 'CW-Fred')
then
(the user U can access service Service1).
```

VII. CE - RSTE Coupling

The key data that must be synchronized between the CE and RSTE layers is the list of policies operating in the system. For each policy in the CE store, there is a corresponding row ('1-1'assignment) in the RSTE Policy Set table. Policies are authored in CE, so a one-way mapping from CE to RSTE is required.

To enable more complex policy decisions to take place at the CE level, other data – such as service and user instances – may be synchronized from the relevant RSTE tables or an external repository. This mechanism is beyond the scope of this paper.

The CE representation of policy described in Section III contains all the necessary features for an equivalent representation in the RSTE schema, as shown in Table X.

TABLE X: Policy Mapping from CE to RSTE

CE Policy	**RSTE field**
Policy ID	PolicyId
Subject	UserAffiliation
Action	Action
Target	ServiceName
Conditions	Service Ownership, Restrictions

As it is mentioned in section III, CE Store is a web application which provides an information processing environment within which Java coded agents can develop and interact with existing CE-based concepts. To enable the communication and thus the management flow from interface to system layer and back, we exploit this ability. The CE agent running in the CE environment, directly reads the list of CE represented policies operating in the system (i.e. policy blocks such as Subject and Action) and utilizing JDBC APIs, updates the data in the RSTE relational database table. The CE agent can also write CE sentences and associated concepts and instances within the CE environment and this way we can establish both top-to-bottom and bottom-to-top communication between high and low layer. When restricted policies are identified at the RSTE layer the agent gets a notification, then negates accordingly the respective conditions of the CE represented policy rule and pushes it back to the user for consideration.

VIII. Conclusion and Future Work

In this paper, we propose a two-layer novel policy based service management model consisting of: the human interface layer, which utilizes the CE controlled natural language environment, and the system level policy layer which is implemented using Restriction Set Theoretic Expressions (RSTE). We briefly describe the capabilities of the interface layer, we provide detailed description of RSTE and we presents the implementation details of the RSTE language using Relational Database and various Relational Algebra operations. We then provide an illustrative scenario which describes the management of a service based system and the service management flow between the two layers. Finally, we describe the ability of the proposed model to report any restricted policies by proposing a policy relaxation to the user by negating policy condition through a backtracking routine when services are not implementable.

In future, we aim to perform detailed evaluation of the proposed model, measuring its efficiency as a holistic multi-layer service management approach. Meaning the performance evaluation of policies' analysis at the high level and he query based, RSTE expressed layer performance evaluation as a decision maker, policy evaluator and policy enforcement enabler when dealing with huge amount, multi-partner authored policies. Finally, the user friendly, CE based policy layer and its rich, ontology based semantics will be utilized for the

development of context aware, sophisticated, interest-based negotiation algorithms, having the human in the loop involved.

ACKNOWLEDGMENT

This research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] S. Y. Shah and B. Szymanski, "Dynamic policy enforcement using restriction set theoretic expressions (rste)," in *IEEE Conference on Military Communications (MilCom), 2014.* IEEE, 2014, p. To appear.

[2] T. Pham, G. H. Cirincione, D. Verma, and G. Pearson, "Intelligence, surveillance, and reconnaissance fusion for coalition operations," in *Information Fusion, 2008 11th International Conference on*. IEEE, 2008, pp. 1–8.

[3] A. Preece, T. Norman, G. de Mel, D. Pizzocaro, M. Sensoy, and T. Pham, "Agilely assigning sensing assets to mission tasks in a coalition context," *IEEE Intelligent Systems*, vol. 28, no. 1, pp. 57–63, 2013.

[4] G. Karjoth, "Access control with ibm tivoli access manager," *ACM Transactions on Information and System Security (TISSEC)*, vol. 6, no. 2, pp. 232–257, 2003.

[5] Ibm research, policy management library. [Online]. Available: https://www.ibm.com/developerworks/community/groups/service\\/html/communityview?communityUuid=ed556565-1d91-4289-94ae-213df1340350

[6] J. Lobo, "Cim simplified policy language (cim-spl)," *Specification DSP0231 v1. 0.0 a, Distributed Management Task Force (DMTF)*, vol. 10, 2007.

[7] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou, and A. L. Lafuente, "Using linear temporal model checking for goal-oriented policy refinement frameworks," in *Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on*. IEEE, 2005, pp. 181–190.

[8] M. Sensoy, T. J. Norman, W. W. Vasconcelos, and K. Sycara, "Owl-polar: A framework for semantic policy representation and reasoning," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 12, pp. 148–160, 2012.

[9] S. A. Chun, V. Atluri, and N. R. Adam, "Using semantics for policy-based web service composition," *Distributed and Parallel Databases*, vol. 18, no. 1, pp. 37–64, 2005.

[10] H. Prakken and M. Sergot, "Dyadic deontic logic and contrary-to-duty obligations," in *Defeasible deontic logic*. Springer, 1997, pp. 223–262.

[11] A. K. Bandara, E. C. Lupu, J. Moffett, and A. Russo, "A goal-based approach to policy refinement," in *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*. IEEE, 2004, pp. 229–239.

[12] R. Dilmaghani, S. Geyik, K. Grueneberg, J. Lobo, S. Y. Shah, B. K. Szymanski, and P. Zerfos, "Policy-aware service composition in sensor networks," in *Services Computing (SCC), 2012 IEEE Ninth International Conference on*. IEEE, 2012, pp. 186–193.

[13] D. Mott, "Summary of controlled english," *ITACS*, 2010.

[14] A. Bernstein and E. Kaufmann, "Gino - a guided input natural language ontology editor," 2006.

[15] G. Hart, M. Johnson, and C. Dolbear, "Rabbit: Developing a control natural language for authoring ontologies," 2008.

[16] T. Kuhn, "An evaluation framework for controlled natural languages," 2010.

[17] C. Parizas, D. Pizzocaro, A. Preece, and P. Zerfos, "Managing isr sharing policies at the network edge using controlled english," 2013.