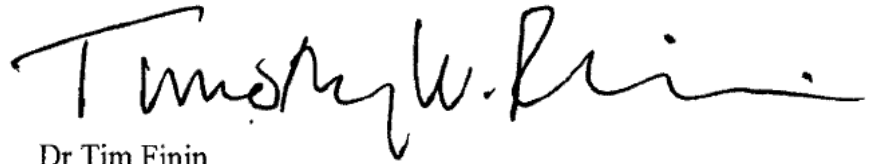


APPROVAL SHEET

Title of Thesis: Group Centric Information Sharing using Hierarchical Models

Name of Candidate: Amit Mahale
Master of Science, 2011

Thesis and Abstract Approved:



Dr Tim Finin
Professor
Computer Science

Date Approved: 6/29/2011

Curriculum Vitae

Name: Amit Mahale

Degree and date to be conferred: M.S in Computer Science, 2011

Secondary education: Karnatak College Dharwad, Karnataka, India, 2004

Collegiate institutions attended:

University of Maryland at Baltimore County, M.S in Computer Science
(Aug 2009 – May 2011).

SDM College of Engineering and Technology, B.E in Information Science
(August 2004 – July 2008).

Major: Computer Science

Professional positions held:

Software Developer Intern, Millennial Media, Baltimore, MD, USA
(June 2010 – Aug 2010).

Programmer Analyst, Cognizant Technology Solutions, Chennai, India
(Nov 2008 – July 2009).

ABSTRACT

Title of Document: GROUP CENTRIC INFORMATION
SHARING USING HIERARCHICAL
MODELS

Amit Mahale, Master of Computer Science, 2011

Directed By: Professor Dr Tim Finin,
Department of Computer Science and
Electrical Engineering

Traditional security policies are often based on the concept of “need to know” and are typified by predefined and often rigid specifications of which principals and roles are pre-authorized to access the information. A recommendations of the 9/11 commission was to find ways to move from this traditional perspective toward one that emphasizes the “need to share”. Ravi Sandhu and his colleagues have developed the Group centric secure information sharing model (gSIS) as a new model that is more adaptable to highly dynamic situations requiring information sharing. We present an implementation of gSIS and demonstrate its usefulness to use-cases in information sharing in social media. Our contributions include the prototype implementation, extension to the model such as hierarchical groups and necessary and sufficient conditions, and the use of the Semantic Web language for representing the central gSIS concepts and associated data. Our framework uses a pragmatic approach of using semantic web technology to represent and reason about the hierarchy and procedural method to compute access decisions relying on the gSIS semantics.

**Group centric information sharing using hierarchical
models**

by

Amit Mahale

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County, in partial fulfillment
of the requirements for the degree of
Master of Science in
Computer Science
2011

© Copyright
Amit Mahale,2011

Dedicated to
Aayee (My Great Grandmother)
(May 10th 1911 – April 22nd 2010)

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my graduate advisor Dr. Tim Finin. His suggestions, motivation and advice were vital in bringing this work to completion. I would like to thank Dr Anupam Joshi for playing an equally important role in advising me towards my work; I am grateful to Dr Laura Zavala for her guidance and suggestion towards the development and improvement of ontology for gSIS. I would like to thank Dr Yelena Yesha for gracefully agreeing to be on my thesis committee. I extend my sincere thanks to Air Force Office of Scientific Research for funding this research under the MURI award FA9550-08-1-0265 (AFOSR). I would also like to thank all my friends at UMBC for their support and encouragement.

Last but not the least; I would like to thank my parents and brother for their support especially my parents for providing the best education at every stage of my life. I am also thankful to my wife for her utmost belief and constant support without which a master's degree would not have been possible.

Table of Contents

Dedication	i
Acknowledgements	ii
List of Tables	v
List of Figures	vi
Chapter 1: Introduction	1
Chapter 2: Background and Related Work	5
2.1 Semantic Web	5
2.2 Group Centric Information Sharing	7
2.2.1 Core properties	7
2.2.1 Group Operation Semantic	9
Chapter 3: System Usecase	13
3.1 Graduate Students Admission	13
3.2 Promotion and Tenure (P&T)	14
3.3 Social Network	15
3.3.1 Incorporating gSIS into Facebook	16
Chapter 4: System Architecture	19
4.1 Group Operation Data	20
4.2 Hierarchy Ontology	20
4.3 Inferred Data	22
4.4 gSIS Ontology	22

4.5 Decision Engine	27
4.6 Model Extension.....	31
4.6.1 Automated Group Membership	31
4.6.2 Automated Document Classification	33
Chapter 5: System Implementation.....	34
Chapter 6: Results	37
Chapter 7: Conclusion and Future Work	42
References.....	40

List of Tables

2.1 Group Operation semantics..	11
---------------------------------------	----

List of Figures

2.1: User Operations (figure courtesy: Ram Krishnan et al [1]).....	10
2.2: Object Operations (figure courtesy: Ram Krishnan et al [1]).....	11
3.1 Adding a Friend.....	16
3.2 Adding a Document/Post.....	17
3.3 Removing a Friend.....	18
3.4 Removing a Document/Post.....	18
4.1 High level system design.....	19
4.2 Hierarchy of Disaster Management Group.....	21
4.3 gSIS Class hierarchy.....	23
4.4 gSIS classes and object properties relations with color codes.....	26
4.5 Strict Join. Strict Add, Strict Leave, Strict Remove operations.....	28
4.6 Liberal Add. Liberal Join, Liberal Remove, Liberal Leave operations.....	29
4.7 Strict Join. Liberal Add, Strict Leave, Liberal Remove operations.....	30
4.8: Liberal Join. Strict Add, Liberal Leave, Strict Remove.....	30
4.9 OWL Model for Automated Group Classification.....	33
5.1 Flowchart of the gSIS Access decision system.....	34
6.1 Query 2: List all the documents that Dr Finin.....	40
6.2 Query 3: List all the users who have access to.....	40
6.3 Query 4: List all the documents that were accessible to users in 1994.....	41
6.4 Query 5: Did Dr Finin ever have access to	42

Chapter 1

INTRODUCTION

The concept of “Need to share” has particularly gained popularity in the aftermath of 9/11 attack in comparison to the traditional Need to Know model.

Taking an example of the US Federal Systems, Each intelligence agency has its own networks and data store that make it difficult to aggregate together the facts and warn of adversaries ahead of time [10]. The inability or unwillingness to share information was recognized as an Intelligence Community weakness by both the 9/11 Commission and the Weapons of Mass Destruction (WMD) Commission [10]. The Need to Share environment is necessary to uncover, respond, and protect against threats.

Secure Information Sharing (SIS) is of prime importance in today’s electronic world.

Its application ranges from highly confidential federal systems to social media application handling user data.

Collaborative systems are not only restricted to federal systems, we can find such systems in day to day life as well. An example for such a collaborative system can be picturized in a university environment, which has a set of systems like Admissions, Library, University Health Service (UHS) sharing data for various purposes like the Admissions department querying the UHS for immunization records prior to student

course registration, this will help the university enforce the policy of student immunization during the start of the semester.

Social Network like Facebook [21] is growing rapidly with currently having more than 500 million registered users. It is a place for Information Sharing wherein people communicate sharing data with others as well as with various third party applications which pull our personal data to provide services on the Facebook platform.

From all the above examples, we understand that data sharing is important; however at the same time protecting the information from unauthorized usage is equally significant.

Traditional access control models do support important SIS aspect though it has not been satisfying for varied domains and requirements of the modern information sharing era. For example Discretionary Access Control (DAC) works on the concept of owner control. Owner has the right to make decision about who can access a particular object. While this is an important SIS aspect, DAC is fundamentally limited in that it controls access only to the original object but not to copies. The lack of constraints on copying information from one file to another makes it difficult to maintain safety policies and verify that safety policies. If objects could be read, one can read and create a copy of this object [7].

Further, DAC is also too fine-grained in practice since the secure information sharing responsibility falls on the owner of the information. The system provides no guidance as to how information can be effectively shared.

Mandatory Access Control or MAC and models such as that of Bell-LaPadula (BLP) assigns security labels to subjects and objects and is based on restricting information flow from more secure classification levels to less secure levels. In BLP, information can only flow from a subject of lower clearance to that of higher clearance and not vice-versa. The intended objective is that of confidentiality of objects at higher security clearance from that of subjects executing at lower clearance which is common in military to allow Generals to see certain information and not Soldiers [11].

The modern concept of Role-Based Access Control (RBAC) [7] is a more generalized model and can be viewed as an evolution of access control to simplify administration in organizations bringing in additional concepts such as hierarchies and constraints. It has also been shown that RBAC is policy neutral in the sense that it can be configured to enforce both DAC and MAC policies [8].

However, As RBAC is too general it does not directly address information sharing does not provide a framework for secure sharing.

Group-Centric sharing differs from other models as it advocates bringing the users and objects together to facilitate sharing by focusing on semantics of group operations.

Our focus in this thesis is to use the concepts of Group Centric Information Sharing [1] and develop ontology's in Web Ontology Language (OWL) [22] and further use these ontology's to build a framework to demonstrate the usefulness of such a system. In this model users and Information (resources/objects) come together in a group to

facilitate sharing. We further extend this model to support hierarchical group. Finally we support our work through a working prototype.

Chapter 2

Background and Related Work

2.1 Semantic Web

The Semantic Web refers to the W3C's vision of the web of linked data. It extends the World Wide Web that enables people to share *content* beyond the boundaries of applications and websites. Semantic Web technologies enable people to create data using RDF, build vocabularies using web ontology language (OWL), write rules and query data stores using SPARQL [8].

The vision of Semantic Web was first articulated by Tim Berners Lee to extend the existing web in which knowledge and data could be published in a form that is easy for computers to understand and reason. This would support more sophisticated software systems that share knowledge, information and data on the Web just as people do by publishing text and multimedia [13].

Under the stewardship of the W3C, a set of languages, protocols and technologies have been developed to partially realize this vision, to enable exploration and experimentation and to support the evolution of those concepts and the technology.

The current set of W3C standards are based on RDF (Lassila et al. 1998), a language that provides a basic capability of specifying graphs with a simple interpretation as

a “semantic network” and serializing them in XML and several other popular Web systems (e.g., JSON). Since it is a graph based representation, RDF data are often reduced to a set of ‘triples’ where each one represents an edge in the graph.

The Web Ontology Language (OWL) (Bechhofer et al. 2004) is a family of knowledge representation languages based on Description Logic (DL) (Baader 2003) with a representation in RDF. OWL supports the specification and use of the ontologies that consist of the terms representing individuals, classes of individuals, properties, and axioms that assert constraints over them. The axioms can be realized as simple assertions (e.g., ‘Woman is a subclass of Person’, ‘hasMother is a property from Person to Woman’, ‘Woman and Man are disjoint’) and also as simple rules. The use of OWL to define policies has several very important advantages that become critical in distributed environments involving coordination across multiple organizations. First, most policy languages define constraints over classes of targets, objects, actions and other constraints (For example, time or location). A substantial part of the development of a policy is often devoted to the precise specification of these classes, e.g., the definition of what counts as a ‘student’ or an ‘entertainment activity’. This is especially important if the policy is shared between multiple organizations that must adhere to or enforce the policy even though they have their own native schemas or data models for the domain in question. Second, OWL is based on description logic, a well understood subset of logic for which powerful and efficient reasoning systems are available. By constraining our use of OWL to the right subset, we can exploit existing OWL reasoners. A third advantage is that OWL’s grounding in logic facilitates the translation of policies expressed in OWL to other

formalisms, either for analysis or for execution. Finally, OWL is designed of and for the Web, making sharing policies and the ontologies they use both natural and easy [4].

2.2 Group Centric Information Sharing

Group centric Information sharing [1,2,3] is a novel concept developed by Ravi Sandhu et al, It envisions bringing the users and objects together in a group to facilitate sharing for a common purpose.

The model focuses on semantics of group operations: Join and Leave for users and Add and Remove for objects, each of which can have two variations namely strict and Liberal. The authors use Linear Temporal Logic (LTL) to characterize the core properties of a group in terms of these operations [1].

We will not dwell into the LTL details and will concentrate on the core gSIS properties followed by the group operation semantics.

2.2.1 Core gSIS Properties

The core properties [1] must be satisfied by any g-SIS specification. The core properties are stated with the assumption that Join, Leave, Add and Remove are the only events that influence authorization in g-SIS. In the future, these properties can be extended to models involving additional aspects (e.g. attributes of users and objects).

1. Persistence Properties: These properties consider the conditions under which authorization may not change.

a. Authorization Persistence: When a user u is authorized to access an object o , it remains so at least until a group event involving u or o occurs.

b. Revocation Persistence: When a user u is not authorized to access an object o , it remains so at least until a group event involving u or o occurs.

A generalized statement of these properties may be "Authorization does not change unless an authorization changing event occurs." With this generalization, we believe persistence property is required of all access control systems.

The following properties are more specifically targeted at g-SIS. They seek to recognize the additional authorizations enabled and disabled by group membership and non-membership respectively.

2. Authorization Provenance: Intuitively, a user will not be authorized to read an object until a point at which both the user and object are simultaneously group members.

Two things can be inferred from the statement, if Authentication holds in a given state then there was an overlapping period of membership between the user and object at least once in the past. Next, authorization to read an object cannot begin for the first time during a user's non-membership period (that is, only joining a group can enable authorization).

3. Bounded Authorization: These properties require that authorizations not Increase during non-membership periods of users and objects (note that authorizations may decrease). Authorizations that hold during the non-membership period of users and object should have held at the time of Leave and Remove respectively.

- a. *Bounded User Authorization:* The set of all objects that a user can access during non-membership period is bounded at Leave time. This set cannot grow until the user re-joins.

The above property states that additional authorizations cannot be granted to a user during non-membership period. Any object that is accessible after Leave should have been authorized at the time of Leave.

- b. *Bounded Object Authorization:* The set of all users who can access a removed object is bounded at Remove time, which cannot grow until re-Add.

4. Availability: Availability specifies the conditions under which authorization must succeed. This property states that after a user joins a group, any object that is added subsequently should be authorized. Obviously, the user should be a current member when the object in question is added.

2.2.2 Group Operation Semantics

The Group operation semantics are the additional properties that are based on specific variations of group operations, these properties define certain group operation semantics that are useful for a variety of applications. All these properties are not required in the development of the system, in any system only a subset of these

properties will be used in accordance to the requirement and semantics of the system, the designer plays a key role in deciding the properties to be used for the system.

Membership Properties characterize the semantics of authorizations enabled when a user joins or an object is added and those which are disabled when a user leaves or an object is removed from the group.

Strict Join (SJ) Vs Liberal Join (LJ): In SJ, the joining user may only access some or all of the objects added after Join time. LJ additionally allows the user to access some or all of the objects that were added prior to join time. Suppose that in figure 2.3.1 the second Join ($u1; g$) is an SJ. Then $u1$ can access $o4$ and $o5$ but cannot access $o2$ and $o3$. If the Join was an LJ instead of SJ, $u1$ can also access $o2$ and $o3$

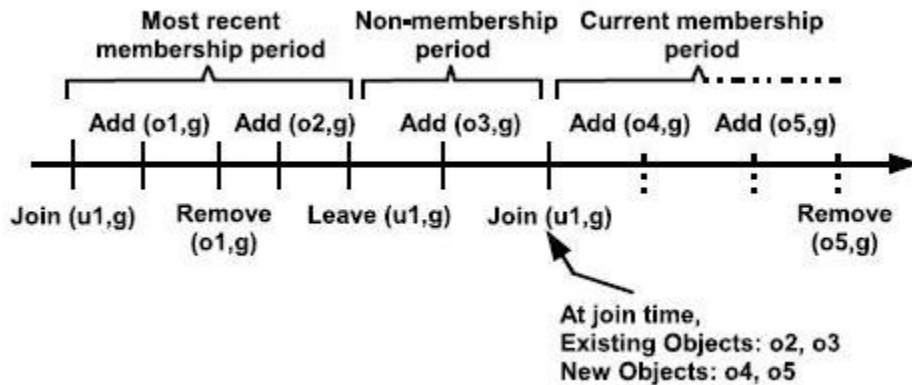


Figure 2.1: User Operations (figure courtesy: Ram Krishnan et al [1])

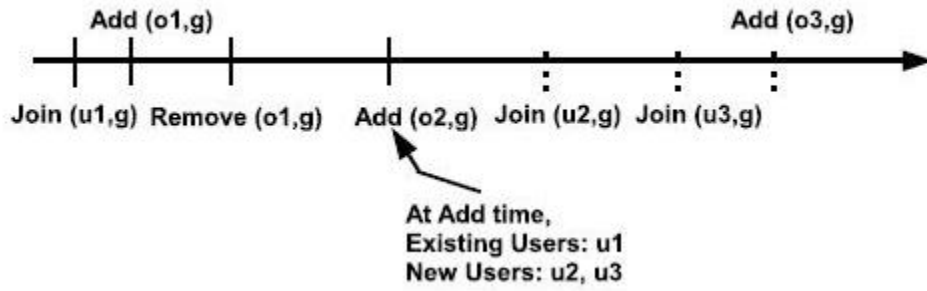


Figure 2.2: Object Operations (figure courtesy: Ram Krishnan et al [1])

Strict Leave (SL) Vs Liberal Leave (LL): In SL, the leaving user loses access to all objects. In LL, the leaving user may retain access to some or all of the objects authorized prior to Leave time. In figure 2.2.1, on SL, $u1$ loses access to all group objects ($o1$ and $o2$) authorized during the membership period. An LL will allow $u1$ to retain access to $o2$ (and possibly $o1$, depending on the type of Remove of $o1$).

Operation	Explanation
Strict Join(SJ)	Only objects added after join time can be accessed
Liberal Join(LJ)	Can access objects added before and after join time
Strict Leave(SL)	Lose access to all objects on leave
Liberal Leave(LL)	Retain access to objects authorized before leave time
Strict Add(SA)	Only users who joined prior to add time can access
Liberal Add(LA)	Users who joined before or after add time may access
Strict Remove(SR)	All users lose access on remove
Liberal Remove(LR)	Users who had access at remove time retain access

Table 2.1: Group Operation semantics (Table courtesy: Ram Krishnan et al [1])

Strict Add (SA) Vs Liberal Add (LA): In SA, the added object may only be accessed by only some or all of the users who joined before Add time. In LA, the added object may also be accessed by some or all of the users that join (e.g., LJ) later. If Add ($o2; g$) in figure 3.2 is an SA, only $u1$ can access the object. Users $u2$ and $u3$, joining later, cannot access this object. But on LA current user $u1$ and future users $u2$ and $u3$ may access $o2$.

Strict Remove (SR) Vs Liberal Remove (LR): In SR, the removed object cannot be accessed by any user. In LR, some or all of the users who had access at Remove time may retain access (of course, users joining later are not allowed to access the removed object this respects the Authorization Provenance core property). In figure 2.3.2, if Remove ($o1; g$) is an SR, every group user (including $u1$) loses access to $o1$. If Remove ($o1; g$) is an LR, $u1$ can continue to access $o1$. However $u2$ and $u3$ will not have access to $o1$.

Chapter 3

SYSTEM USE CASE

Secure Information Sharing (SIS) or sharing information while protecting is necessary. Use cases for SIS vary from applications like secure meeting room to collaboration between organizations and social networking application handling user interaction with an expectation of security and privacy.

3.1 Graduate Student Admissions: Graduate admissions [17] is a process where in the graduate applications are scrutinized by a group of faculty members from the department. The group consists of a mix of senior professors, Associate professors and senior graduate students working towards the completion of masters and PhD.

Prospective graduate students send their applications to the department for evaluation and the committee weighs the credibility of the applicant based on multiple factors and makes a decision about his admission.

The gSIS model facilitates and promotes the process of information sharing among the various committee members. As our model supports hierarchical groups handling groups of Professors, Associate professors and grad students within the Graduate student admissions group is simplified.

To implement this model, we have to enforce the following gSIS operations

- Enforce users to Join the group though 'Liberal Join', This would make sure that in additional to the applications added to for this academic year, members

can also access previous applications to get better understanding of university's selection pattern.

- Add the application documents with 'Liberal Add' so that even committee member joining the committee at a later point of time can access the applications.
- If a member leaves the group, then use 'Strict Leave', so that he/she loses all access to the documents of the Group.
- If the documents are to be removed from the group for some reason, then use the 'Liberal Remove', this will ensure that the members present during the review of that particular application have access to the removed documents for analysis purpose.

3.2 Promotion and Tenure Committee (P&T): A promotion and tenure committee [16] consists of a group of full professors (tenured) who decide on the fate of an Associate professor under consideration for tenure.

The promotion and tenure committee resembles the group centric information sharing as the group shares information towards a single goal, the goal being decision over tenure, Also the access level of the members of the group varies from individual to another. This is mainly dependent on his seniority in the group (Join timestamp of the member).A senior member of the group can check the tenure documents of his fellow junior group members but not the vice-versa. This serves as an excellent use case for our model of group centric information sharing.

To implement gSIS for this use case, we would have to use the following group operations,

- Enforce users to Join the group through ‘Strict Join’, This would make sure that the users can access only the documents added after their join time.
- Add the P&T documents with ‘Strict Add’ so that only users joining prior to Add time can access the documents.
- If a tenured professor leaves the group, then use ‘Strict Leave’, so that he/she loses all access to the documents of the Group.
- If the documents are to be removed from the group for some reason, then use the ‘Strict Remove’, this will ensure that none of the members have access to the removed documents.

3.3 Social Media Application: Social Media platforms like Facebook handle user profile information ranging from basic information to interests and social network data. Currently when Bob becomes a friend of Alice on Facebook, Bob gets access to all the personal information as well as the content (from Facebook Wall) that Alice had shared earlier with her friends. Thus unintentionally sharing the data with Bob that she has never intended to do so, this can cause serious privacy infringement [11, 15] to Alice.

This issue can be fixed by using the gSIS operations semantics while sharing information and adding new friends to our existing list of friends.

Let us dwell into the details of gSIS operators and its semantic in social network

- **Strict Join:** if Alice adds a new friend Bob to her friend list through Strict Join, then Bob will not be able to access any of the posts (In this scenario posts and documents are used interchangeably) shared by Alice prior to his Join time. Thus Bob will not be able to spy about Alice's online behavior
- **Liberal Join:** In addition to allowing access to new documents, Liberal Join would allow Bob to access posts that Alice shared prior to his join time through Liberal Add.
- **Strict Add:** Alice should use this operation, if she wants to share the post with current set of friends and protect from her future friends.
- **Liberal Add:** This post can be accessed by current friends as well as new friends who join at a later point of time through Liberal Add.

If we carefully give a thought about the current Facebook model, we can understand that it works on lines of Liberal Join for adding new friends to our list and Liberal Add while posting documents.

About the delete and Remove options, Facebook currently emulates the Strict Leave and Strict Remove semantics of gSIS.

3.3.1 Incorporating gSIS into Facebook

➤ Adding a Friend

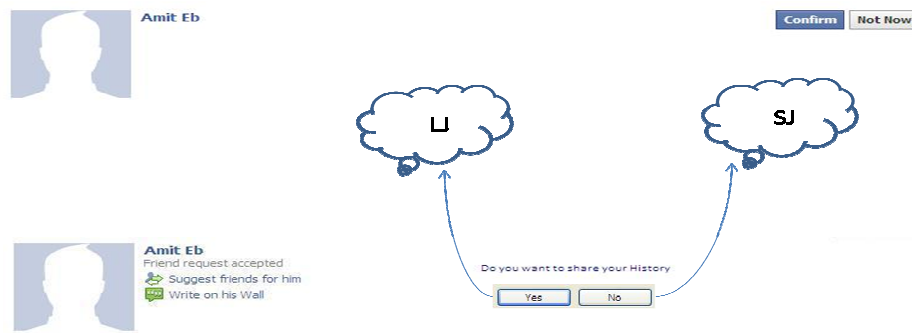


Figure 2.1: Adding a Friend

After adding a friend to the list, The user preference can be asked as illustrated above.

Sharing history with the newly added friend would mean liberally adding the friend whereas preferring not to share the history would mean strictly adding the friend.

➤ **Adding a Post**

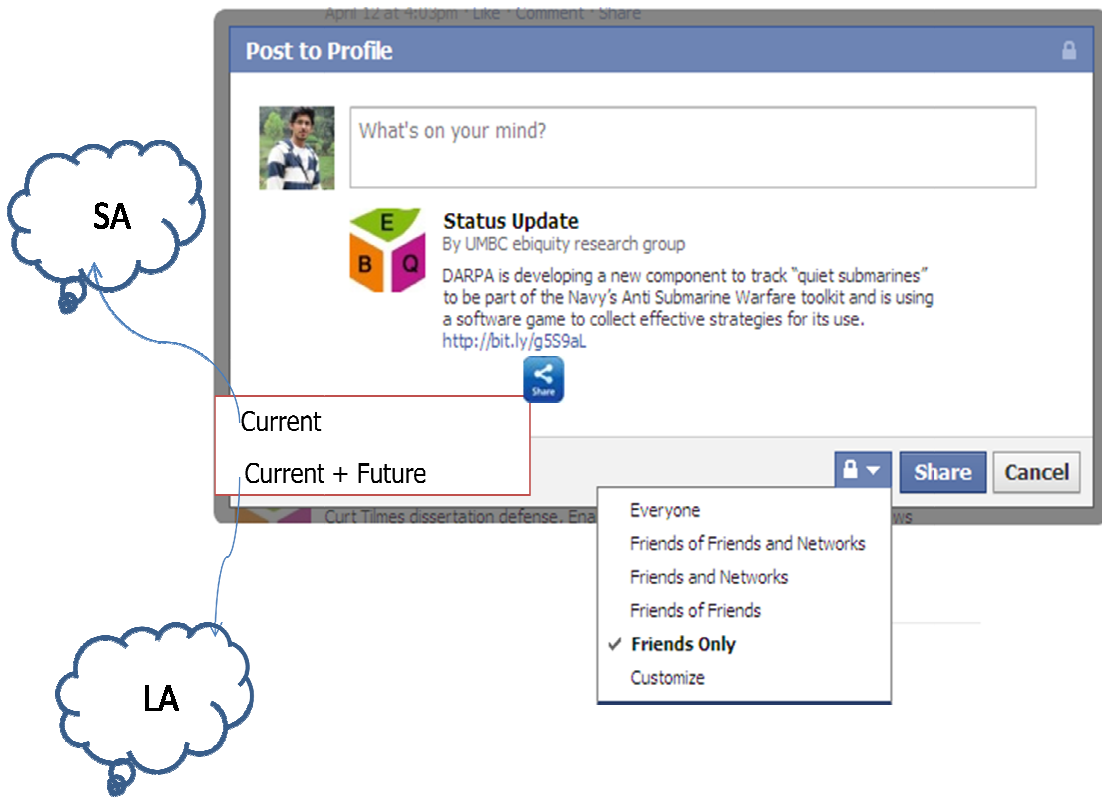


Figure 3.2: Adding a Document

While adding a post, the user can have a option ' Share this post with my current set of friends' v/s 'Share the post with my current and future set of friends', the former meaning strictly adding the post to the profile while the later liberally adding the post.

➤ **Removing a Friend**

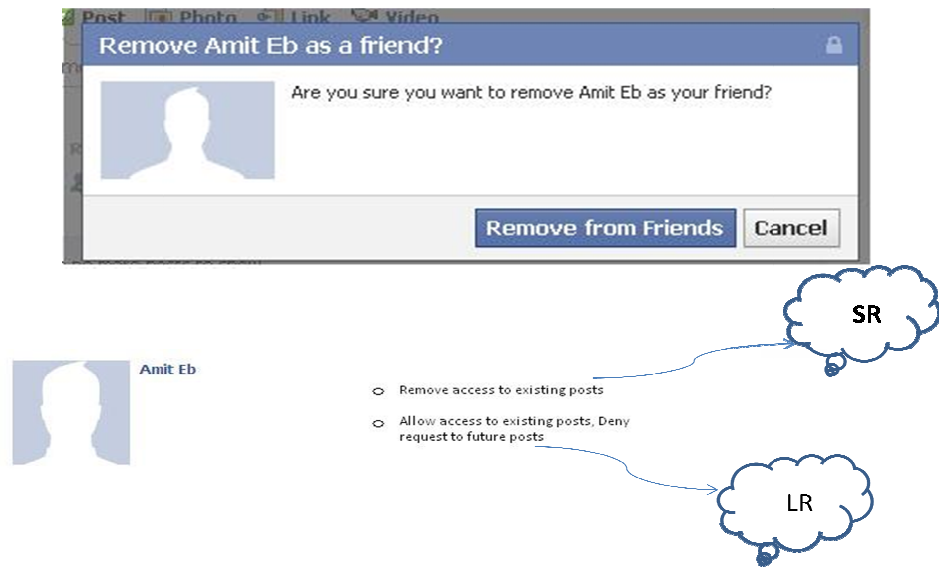


Figure 3.3: Removing a Friend

➤ **Removing a Post**



Figure 3.3: Removing a document(Post)

Chapter 4

SYSTEM ARCHITECTURE

The high level system design (Fig 4.1) demonstrates the Group Centric Information sharing setup

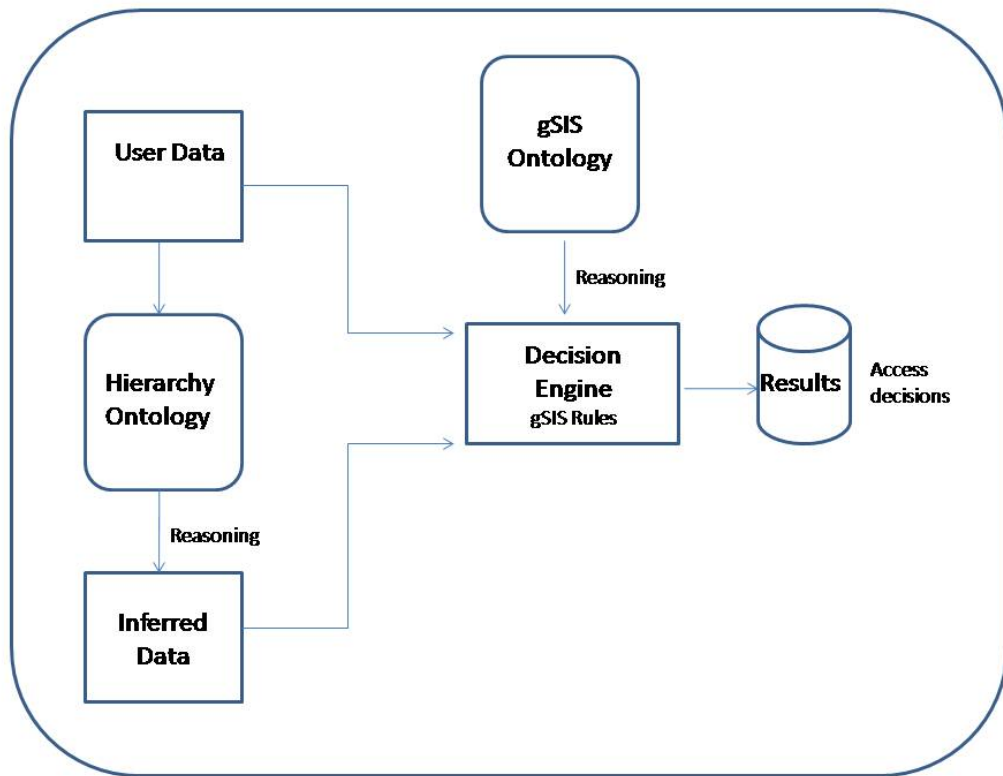


Figure 4.1 High level system design

The system is built to make access decisions in a group centric information sharing environment.

Group-Centric sharing brings in the users and objects together to facilitate sharing by focusing on semantics of group operations. User's join the group through join operation and leave the group through leave operation. The join and leave operations further have strict and liberal flavors which are explained in the background section.

Similarly, the documents are added to the group through Add operation and removed using the Remove operation. Even Add/Remove has strict and liberal variations analogous to the Join/Remove.

We will now discuss the system component in detail

4.1 Group Operation data

The Group operation data is the data about the group members and their group operations. Every member of the group either user or document is identified by a unique id. There is no restriction on the number of transactions a member can have with the group. In other words a group user can join and leave the group multiple numbers of times without hurting the core gSIS properties.

4.2 Hierarchy Ontology

The hierarchy ontology is responsible for inferring the groups that the group member belongs to. In real life scenario, groups may be created with hierarchy in

mind. For example consider a hierarchy of Professor, Asst Professor and Lab Instructor.

Thus a user added to a Professor group should by default have access to the documents added to Asst Professor and Lab Instructor group.

Thus the use of hierarchy ontology along with a reasoner reduces manual work and automates the task of inferring groups that the user represents.

Another example can be quoted of a Disaster management group

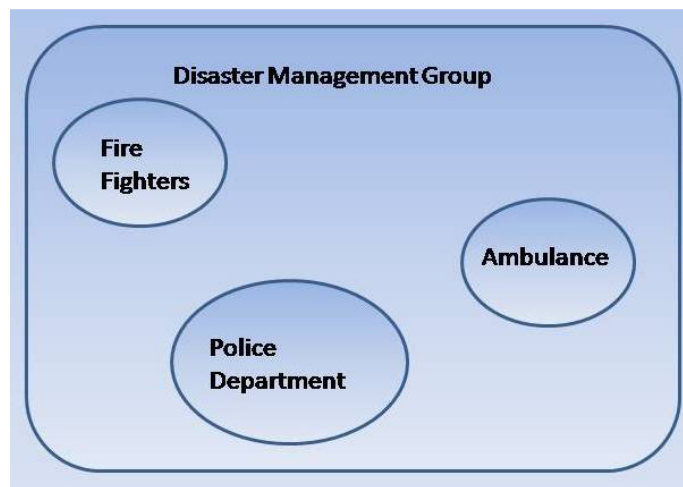


Figure 4.2 Hierarchy of Disaster Management Group

From the figure, we understand that the Disaster Management Group(DMG) is a large group comprising of the fire fighters, Police department and Ambulance who have access to a particular subset of documents of the Disaster management group. The DMG as a whole can access any of the documents of Fire fighters, Police Dept and Ambulance but not vice versa.

In such a hierarchical setup, the documents added to Police department should be accessible to the DMG as it is a super groups; this fact is true for other groups as well.

Thus the use of hierarchy ontology helps to automate the task of inferring additional groups and facilitating information sharing for hierarchical groups.

The hierarchical groups can be represented in OWL using the property :subclassOf
<RoleName> gSIS:subclassOf <SuperRoleName>.

E.g.: PoliceDepartment gSIS:subclassOf :DMG

4.3 Inferred Data

The RDFS reasoner is used to infer additional group data using the hierarchy ontology. The inferred data is then stored in a data store along with the group data and is used to make access decisions in further steps.

4.4 gSIS Ontology

Our work is based on the theory of Group-Centric Secure Information Sharing (g-SIS) described in [1], our focus is on creating ontology to represent the concepts in the g-SIS model using OWL.

Our gSIS ontology primarily consists of four classes: Person, Document, Group and Action.

The Action class is further divided into the Join, Leave, Add, and Remove each of which further have Strict and Liberal variations. The Join and Leave actions are used to represent the fact that a Person can join or leave a Group. Similarly, the Add and Remove actions represent the fact that a Document can be added or removed from a Group.

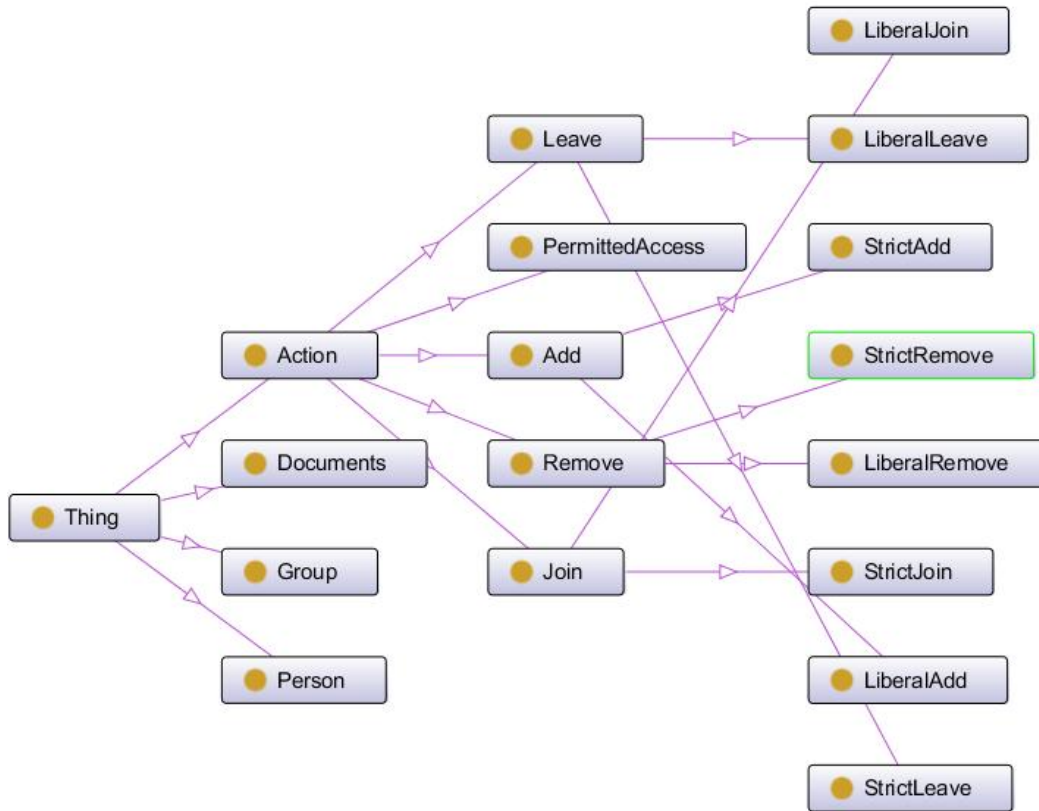


Figure 4.3: gSIS classes from top level to bottom

Actions can be allowed or denied depending on a few conditions on the time and a combination on the variations (Strict or Liberal). For example, a Person can access a Document in a Group if the person joined the group before the document was added and the person joined with a strict joined and the document was added with a strict add.

In order to represent the fact that an action is allowed (or not), we have created a PermittedAction class, which is a subclass of Action which is used by the person and document classes to check if the action is permitted according to the group semantics.

Here is an example of an Action class declaration in owl

```
:Action rdf:type owl:Class .  
:Add rdf:type owl:Class ;  
rdfs:subClassOf :Action .
```

Further the ontology has object properties like ;hasDocument, :hasGroup, :hasPerson

The :hasDocument property is defined in owl as follows

```
:hasDocument rdf:type owl:InverseFunctionalProperty  
, owl:ObjectProperty ;  
rdfs:domain :Add, :PermittedAccess, :Remove .  
rdfs:range :Documents ;
```

The domain for :hasDocument vary from Add, Remove and PermittedAccess and the range is restricted to the Documents. This will allow an individual instance of Add, Remove or Permittedaccess to link to Document.

The :hasGroup property is defined in owl as follows

```
:hasGroup rdf:type owl:ObjectProperty ;  
rdfs:domain :Add, :Join, :Leave, :Remove ;  
rdfs:range :Group.
```

The :hasPerson property is defined in OWL as follows

```
:hasPerson rdf:type owl:ObjectProperty ;
```

```
rdfs:domain :Join , :Leave ,:PermittedAccess ;  
rdfs:range :Person .
```

The ontology also has data property named :hasTimestamp which manages the timestamp for the group operations like adding, removing documents and joining, leaving the groups for members.

Let us walk through a simple example demonstrating the usage of gSIS ontology

```
:SJ1rdf:type :StrictJoin ,  
owl:NamedIndividual .  
:  
:Amitrdf:type :Person ,  
owl:NamedIndividual .  
:  
:Ebiquityrdf:type :Group ,  
owl:NamedIndividual .
```

```
:SJ1 :hasPerson :Amit  
:  
:hasTimestamp XXXX  
:  
:hasGroup :Ebiquity
```

The relationship between the classes and the object properties is represented in the next graph

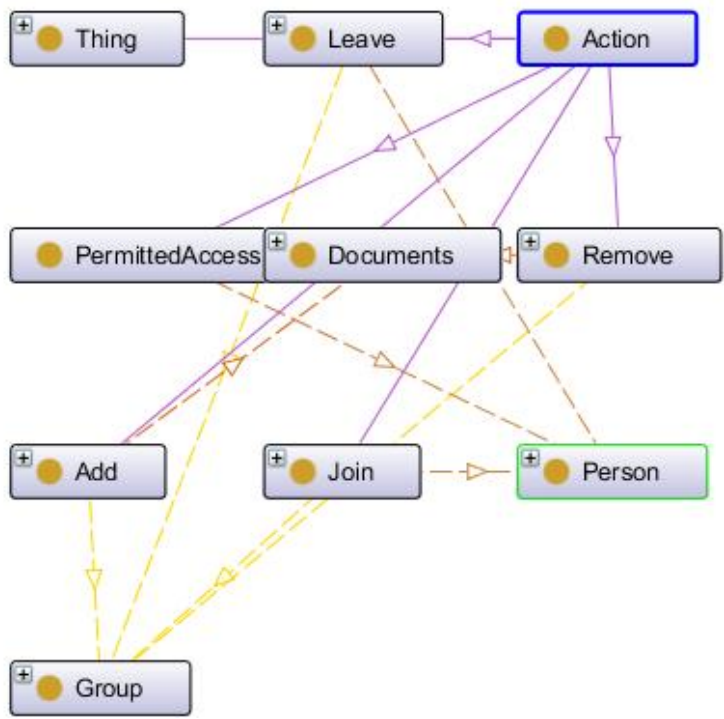


Figure 4.4: gSIS classes and object properties relations with color codes

<input checked="" type="checkbox"/>	— has subclass
<input checked="" type="checkbox"/>	— hasDocument (Domain>Range)
<input checked="" type="checkbox"/>	— hasGroup (Domain>Range)
<input checked="" type="checkbox"/>	— hasPerson (Domain>Range)

4.5 Decision Engine

The Decision engine is the central system of the gSIS model; the rules pertaining to the working of gSIS are encoded in this module.

gSIS is governed by a number of parameter's which control the access decision between the user and the document. Along with the group operations the timestamp's associated with the Join, Leave, Add, and Remove are critical in making access decisions to documents added to the group.

After analyzing the concepts, we have come up with a 16 combination of events that can occur in a group centric information sharing environment and modeled our rules to accommodate all possible interactions

Before jumping into the rules, let us briefly touch upon the basics axioms of gSIS,

- i. Every user and document is associated with at least one group.
- ii. Multiple groups may exist.
- iii. Groups may further be hierarchical.
- iv. A user may join and leave the group multiple number of times.
- v. A document may be added and removed from the group multiple number of times.
- vi. The access decision of a user to a document depends on multiple factors like Join type, Add type and the timestamps associated.

Let us consider the following scenarios

4.5.1 Strict Join, Strict Add, Strict Leave, Strict Remove

In this scenario the user joins the group through Strict Join and leaves the group through Strict Leave, whereas the documents are added through Strict Add and removed through Strict Remove

From the definition [1],

- Strict Join: Only objects added after join time can be accessed.
- Strict Add: Only users who joined prior to add time can access.
- Strict leave: Lose access to all objects on leave.
- Strict Remove: All users lose access on remove.

Let U_j & U_L be the User Join and Leave time and D_A & D_R be the Document Add and Remove time

Then let us plot a simple example with these details on the time line.



Figure 4.5: Strict Join, Strict Add, Strict Leave, Strict Remove operations

From the timeline and the operations semantics, we find the document can be accessed by the designated user from the fig between

$$\text{Access time} = [D_A - \text{Min}(U_L, D_R)]$$

4.5.2 Liberal Join, Liberal Add, Liberal Leave, Liberal Remove

From the definition [1],

- Liberal Join: Can access objects added before and after join time.
- Liberal Add: Users who joined before and after add time can access.
- Liberal leave: Retain access to objects authorized before leave time.
- Liberal Remove: Users who had access at remove time retain access.

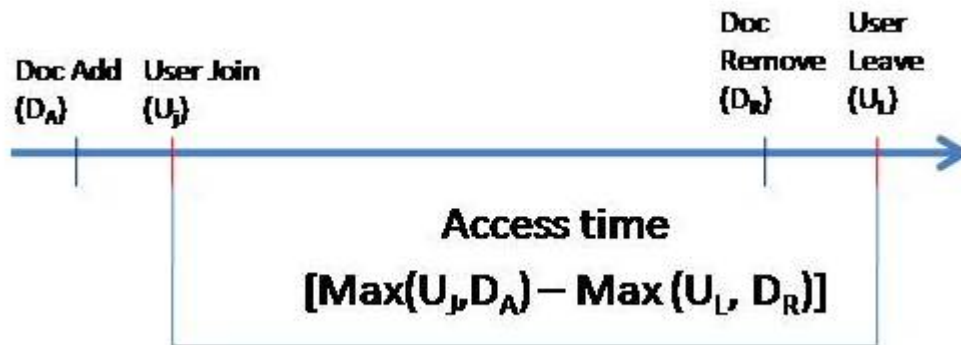


Figure 4.6 Liberal Add, Liberal Join, Liberal Remove, Liberal Leave operations

From the timeline and the operations semantics, we find the document can be accessed by the designated user from the fig between

$$\text{Access time} = [\text{Max}(D_A, U_J) - \text{Max}(U_L, D_R)]$$

4.5.2.1 Strict Join, Liberal Add, Strict Leave, Liberal Remove

Plotting a simple timeline for this scenario, we have

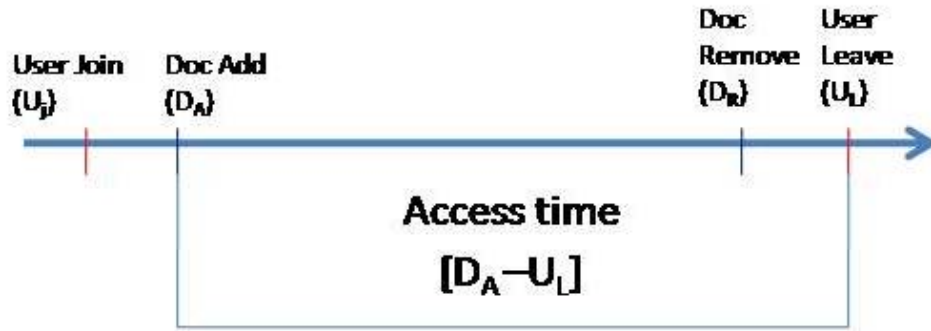


Figure 4.7 Strict Join, Liberal Add, Strict Leave, Liberal Remove operations

From the timeline and the operations semantics, we find the document can be accessed by the designated user from the fig between

$$\text{Access time} = [D_A - U_L]$$

4.5.3 Liberal Join, Strict Add, Liberal Leave, Strict Remove

Plotting a simple timeline for this scenario, we have



Figure 4.8: Liberal Join, Strict Add, Liberal Leave, Strict Remove

From the timeline and the operations semantics, we find the document can be accessed by the designated user from the fig between

$$\text{Access time} = [D_A - D_R]$$

From the above scenario's we can understand that the access decision is dependent on multiple factors like operation type (Join, Leave, Add, Remove), operation time (timestamps associated with the operation) and group membership. As representation of all the mentioned parameter's and constructing the rule becomes overly tedious and complex to handle in OWL [5], we propose an alternative approach for the purpose of building a working prototype of the gSIS framework. The prototype consists of a decision engine developed using the Java environment and an access decision algorithm which takes into account all the above mentioned parameters and provides a fast and intuitive access decision system. The algorithm and implementation details are covered in detail in the next chapter.

4.6 Model Extensions

Group management becomes a tedious task when the number of groups and members increase. One way to manage this process is to automate group membership. As we are using OWL to represent our system, we can use the OWL's Necessary and sufficient conditions to manage group membership

4.6.1 Automated Group Membership

The process of adding users to the relevant group can be a tedious process especially when the users belong to multiple overlapping groups. This process of membership can be automated by defining the necessary and sufficient (N&S) conditions for each group and modeling the same using OWL.

As an example, we can consider the group to be 'UMBC CS Tenure group' and the membership requirement for this group is

- She/he is a Full Professor

- A Professor @ UMBC.
- Faculty in the CS Department.

These conditions can be represented in OWL using the N&S conditions

```

<owl:Class rdf:ID="Tenure_Committee_UMBC_CS">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#CS"/>
      <owl:onProperty>
        <owl:ObjectProperty
rdf:ID="hasDepartmentName"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty
rdf:ID="hasUniversityName"/>
          </owl:onProperty>
          <owl:allValuesFrom
rdf:resource="#UMBC"/>
        </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty>
            <owl:ObjectProperty rdf:ID="hasRank"/>
          </owl:onProperty>
          <owl:allValuesFrom>
            <owl:Class rdf:ID="Full__Professor"/>
          </owl:allValuesFrom>
        </owl:Restriction>
      </rdfs:subClassOf>

```

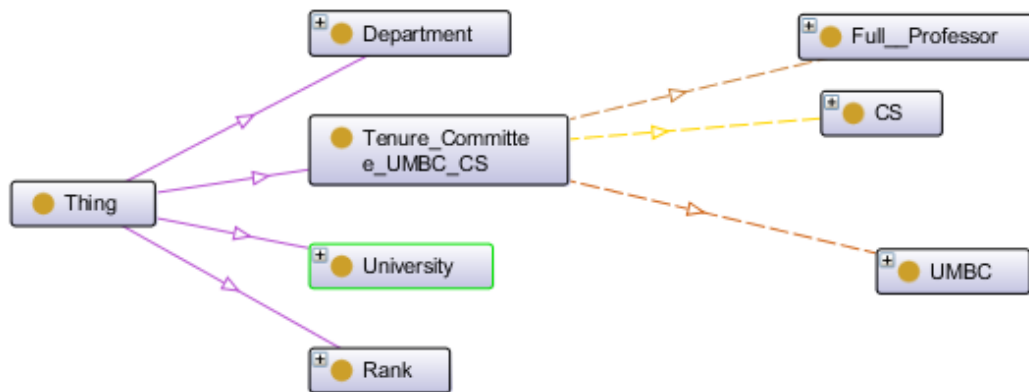


Figure 4.9 OWL Model for Automated Group Membership

Thus once a member with the above characteristics is added to the system then they are automatically classified as the members of the UMBC CS Tenure group.

4.6.2 Automated document classification

This feature is especially useful for federal applications which deal with classified documents. It is crucial that only the right set of people get access to the confidential documents.

Classified information is sensitive information to which access is restricted by law or regulation to particular groups of persons. Documents are usually classified as Top Secret, Secret, Confidential, Restricted and Unclassified.

Groups can be governed by policies on the type of documents that would be a part of the group. For example the 'War room group' should have access to all the Top Secret documents and 'Air Force Group' can have access to documents which belong to the 'Air Force domain' and are classified as 'Top Secret'. These rules can be enforced by using OWL's Necessary and sufficient conditions and the process of document classification can be automated.

Chapter 5

SYSTEM IMPLEMENTATION

Let us look into the flowchart of our access decision system

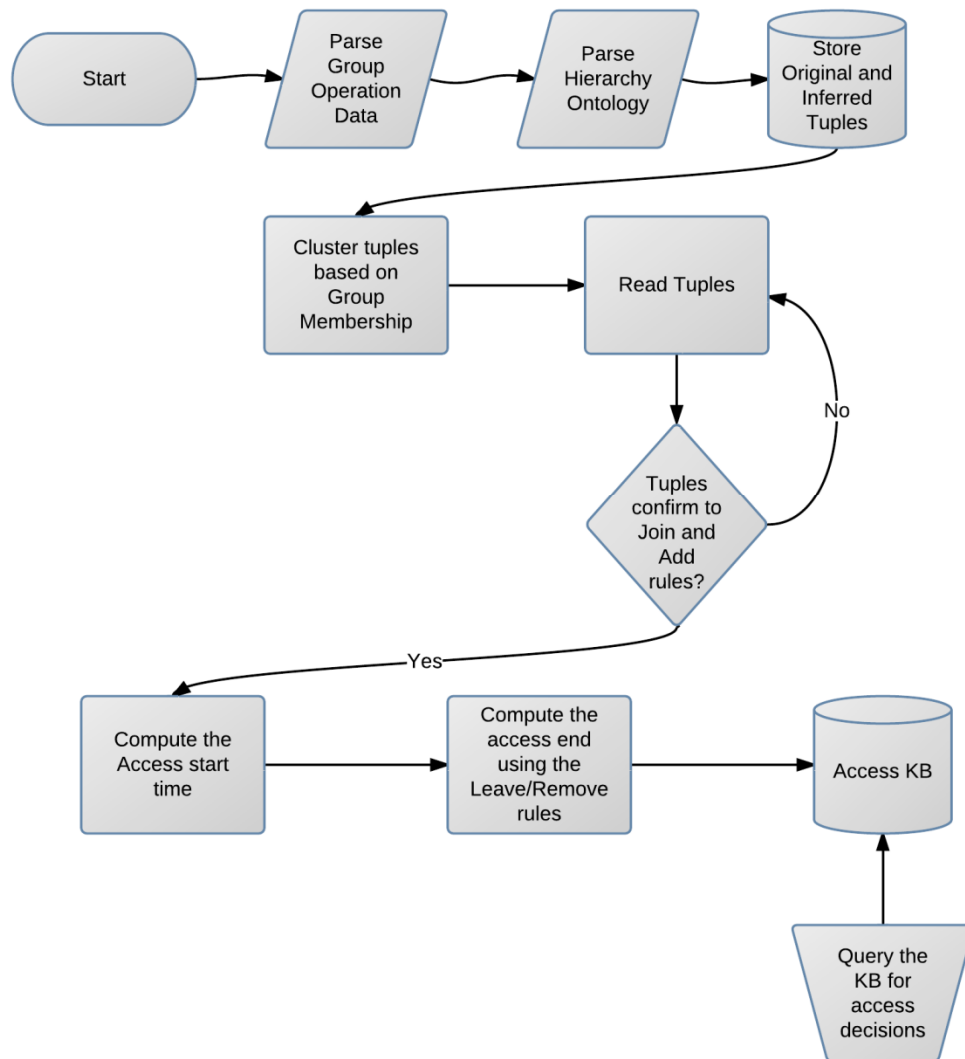


Figure 5.1 Flowchart of the gSIS Access decision system

The access decision algorithm consists of the following stages,

- i. Read the file and parse the Group Membership details.
- ii. Read the hierarchy ontology file and generate the additional tuples using a reasoner by using the original Group membership data.
- iii. Store the original and inferred tuples.
- iv. Cluster the tuples in accordance to their group membership.
- v. Clustered tuples are read pair wise consisting of user and document membership details.
- vi. The next stage is to compute access interval between every user and document of the group. The precomputed access intervals will greatly improve the system's readiness to handle any number of access decision queries.
 - a. The pair is tested against the gSIS Join and Add semantics, if true
 - i. The access start time is computed, [computation details are explained in the previous section and depend on the type and timestamp of the operation].
 - ii. The access end time is computed depending on the Leave and Remove semantics.
 - iii. The generated access interval tuples are stored in the following format.

<userid>,<docid>,<start_time>,<end_time>
- vii. The system can now accept queries about access decision between any user and document that is/was a part of the group.

A sample query would be “Does Amit has access to ppt at time stamp X” and the system would look into the Access KB and answer the query.

Whenever the group membership changes, the system recomputes the access intervals to maintains the Access KB up to date.

Chapter 6

RESULTS

6.1 Validation

Let us visualize the scenario of the Promotion & Tenure (P&T) committee use case in our prototype.

The group membership information is the input to this prototype and is the format

```
<user_id>,<join_time>,<join_type>,<leave_time>,<leave_type>,<group_name>
```

Sample data:

```
finin,1990,SJ,2011,SL,tenure_committee  
joshi,1998,SJ,2011,SL,tenure_committee  
nicholas,1995,SJ,2011,SL,tenure_committee  
yesha,1993,SJ,2011,SL,tenure_committee  
dejardens,2001,SJ,2011,SL,tenure_committee  
oates,2003,SJ,2011,SL,tenure_committee  
Andrew,2010,SJ,2011,SL,asso_prof_committee
```

Data represents the committee member (users) of the group. For example the first tuple about Dr Finin says that, The user finin joined the tenure_committee in 1990 through Strict Join(SJ) and is still the part of the committee, For the purpose of programmatic computation we set the leave date to current year.

Similarly documents are added to the group in the format

<doc_id>, <Add_time>, <Add_type>, <Remove_time>, <Remove_type>
>, <group_name>

finindoc,1990,SA,2011,SR,tenure_committee

joshidoc,1998,SA,2011,SR,tenure_committee

nicholasdoc,1995,SA,2011,SR,tenure_committee

yeshadoc,1993,SA,2011,SR,tenure_committee

dejardensdoc,2001,SA,2011,SR,tenure_committee

oatesdoc,2003,SA,2011,SR,tenure_committee

Andrewdoc,2010,SA,2011,SR,asso_prof_committee

Every tenured member of the committee has a tenure document associated with them that is a part of the tenure_committee.

In this sample example, Andrewdoc is the document of Dr Andrew who is been considered for tenure and this document is a part of Associate professor group named asso_prof_committee.

Let us walk through the process, in which the access intervals are computed,

1. Read the Group operations data file
2. In the next step, we read the hierarchy ontology file and generate the additional tuples using a rdfls reasoner and the original Group membership data.

In this case the hierarchy ontology file consists of two classes, the tenure_committee' class and the sub class 'asso_prof_class', which implies that

members of tenure_committee are also a part of the asso_prof_class and these tuples are inferred and stored by the reasoner.

3. In the next part, the tuples are read pair wise and tested for correctness in accordance to gSIS properties and the access start and end time is computed.

This information is stored in the knowledge base in the format

<user_id> , <doc_id> , <start_time> , <end_time> ,

<group_name>

finin , nicholasdoc , 1995 , 2011 , tenure_committee

finin , joshidoc , 1998 , 2011 , tenure_committee

finin , finindoc , 1990 , 2011 , tenure_committee

finin , oatesdoc , 2003 , 2011 , tenure_committee

finin , Andrewdoc , 2010 , 2011 , asso_prof_committee

finin , dejardensdoc , 2001 , 2011 , tenure_committee

finin , yeshadoc , 1993 , 2011 , tenure_committee

The sample output is only for representation purpose and contains tuples only for the member 'Finin', However the actual output access intervals are computed for all group members and stored in the knowledge base.

The knowledge base is updated whenever group membership changes to maintain consistency.

Once the knowledge base is ready it can answer queries of the format

- Query 1: Does Dr Finin have access to Dr Joshi's Tenure file in 2005?

Access Granted

- Query 2: List all the documents that Dr Finin had access to

```
*****
finin,nicholasdoc,1995,2011,tenure_committee
finin,joshidoc,1998,2011,tenure_committee
finin,finindoc,1990,2011,tenure_committee
finin,oatesdoc,2003,2011,tenure_committee
finin,Andrewdoc,2010,2011,asso_prof_committee
finin,dejardensdoc,2001,2011,tenure_committee
finin,yeshadoc,1993,2011,tenure_committee
*****
```

Figure 6.1 Query 2: List all the documents that Dr Finin had access to

- Query 3: List all the users who have access to 'Andrewdoc'

[Andrew is an Assistant Prof and under consideration for tenure]

```
*****
joshi,Andrewdoc,2010,2011,asso_prof_committee
finin,Andrewdoc,2010,2011,asso_prof_committee
oates,Andrewdoc,2010,2011,asso_prof_committee
dejardens,Andrewdoc,2010,2011,asso_prof_committee
yesha,Andrewdoc,2010,2011,asso_prof_committee
nicholas,Andrewdoc,2010,2011,asso_prof_committee
*****
```

Figure 6.2 Query 3: List all the users who have access to 'Andrewdoc'

- Query 4: List all the documents that were accessible to users in 1994

```
*****
finin,finindoc,1990,2011,tenure_committee
finin,yeshadoc,1993,2011,tenure_committee
yesha,yeshadoc,1993,2011,tenure_committee
*****
```

Figure 6.3 Query 4: List all the documents that were accessible to users in 1994

- Query 5: Did Dr Finin ever have access to Nicholasdoc?

```
*****
finin,nicholasdoc,1995,2011,tenure_committee
*****
```

Figure 6.4 Query 5: Did Dr Finin ever have access to Nicholasdoc

As the access intervals are pre computed, the query execution time is less and thus it increases the responsiveness of the system. Such a scheme is efficient when the group membership is comparatively stable and the number of access queries to be answered at any point of time is large i.e.

At time t, No of group operations \ll No of access queries.

Chapter 7

CONCLUSION AND FUTURE WORK

In today's world, there is a serious need for Information sharing model, On these lines we have made an effort to demonstrate the worthiness of gSIS model in handling real world scenario's.

We have presented a framework for gSIS that promotes information sharing, our focus also relied on modeling hierarchical groups and automating group membership using semantic web.

The usefulness of gSIS model has also been demonstrated in real world applications like Graduate Student admissions, P & T committee and Social Media applications.

In this thesis, we have focused on the operational semantics of gSIS model without taking into consideration the administrative operation. We realize that the administrative model is indeed necessary and is required for the gSIS model to grow as a whole. Our next immediate task would be to work on this aspect of gSIS.

REFERENCES

[1] Ram Krishnan, Ravi Sandhu, Jianwei Niu and William Winsborough, [Foundations for Group-Centric Secure Information Sharing Models](#). Proc. 14th ACM Symposium on Access Control Models and Technologies (SACMAT), Stresa, Italy, June 3-5, 2009, pages 115-124.

[2] Ram Krishnan, Ravi Sandhu, Jianwei Niu and William Winsborough, [Towards a Framework for Group-Centric Secure Collaboration](#). In Proc. 5th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), Crystal City, Virginia, November 11-14, 2009, pages 1-10.

[3] Ravi Sandhu, Ram Krishnan, Jianwei Niu and William Winsborough, [Group-Centric Models for Secure and Agile Information Sharing](#). In Proceedings 5th International Conference, on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2010, St. Petersburg, Russia, September 8-10, 2010, pages 55-69. Published as Springer Lecture Notes in Computer Science Vol. 6258, Computer Network Security (Igor Kottenko and Victor Skormin, editors), 2010.

[4] T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham, ROWLBAC - Representing Role Based Access Control in OWL, Proceedings of the 13th ACM symposium on Access Control Models and Technologies, ACM Press New York, June 2008.

- [5] Anne Cregan, Malgorzata Mochol, Denny Vrandecic, Sean Bechhofer [*Pushing the limits of OWL, Rules and Protégé. A simple example*](#) Workshop - OWL: Experiences and Directions (OWLED-2005), Galway, Ireland, November 2005
- [6] R. Sandhu et al, [Role-Based Access Control Models](#), IEEE Computer, 29(2):38-47, Feb 1996, [Google Scholar Search](#)
- [7] R. Sandhu and P. Samarati, [Access Control: Principles and Practice](#), IEEE Communications, 32(9): 40-48, Sept. 1994, [Google Scholar Search](#)
- [8] Semantic web: <http://www.w3.org/2001/sw/>
- [9] Bechhofer, S.; van Harmelen, F.; Hendler, J.; Horrocks, I.; McGuinness, D.; Patel-Schneider, P.; and Stein, L. 2004. Owl web ontology language reference. w3c recommendation.
- [10] United States Intelligence community ‘INFORMATION SHARING STRATEGY’, Office of the Director of National Intelligence, http://www.dni.gov/reports/IC_Information_Sharing_Strategy.pdf
- [11] Jones, H., and Soltren, J. 2005. Facebook: Threats to privacy.

- [12] Ezedin Barka and Ravi Sandhu, [Role-Based Delegation Model/ Hierarchical Roles \(RBDM1\)](#), ACSAC 2004.
- [13] W3C. 2010. Primer: Getting into rdf and semantic web using n3 (<http://www.w3.org/2000/10/swap/primer>).
- [14] Lujun Fang, Heedo Kim, Kristen LeFevre, and Aaron Tami, A Privacy Recommendation Wizard for Users of Social Networking Sites, CCS 10, 2010
- [15] Ralph Gross, Alessandro Acquisti, Information Revelation and Privacy in Online Social Networks, ACM Workshop on Privacy in the Electronic Society (WPES),2005
- [16] Promotion and Tenure in UMBC
http://www.umbc.edu/provost/Faculty_Handbook/section6.pdf
- [17] Karthik Raghunathan, Demystifying the American Graduate Admissions Process, 2010
- [18] Hierarchical Groups, http://en.wikipedia.org/wiki/Hierarchical_organization
- [19] James Hollenbach, Joe Presbrey and Tim Berners-Lee, Using RDF Metadata To Enable Access Control on the Social Semantic Web, Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK 2009).

- [20] Sandford Bessler and Joachim Zeiss, Semantic modelling of policies for contextaware services, 17th Wireless World Research Forum (WWRF'06)
- [21] Facebook, <https://www.facebook.com/>
- [22] Web Ontology Language, <http://www.w3.org/TR/owl-ref/>
- [23] Weitzner, D. J.; Hendler, J.; Berners-lee, T.; and Connolly, D. 2004. Creating a policy-aware web: Discretionary, rule-based access for the world wide web. In in Elena Ferrari and Bhavani Thuraisingham, editors, Web and Information Security. IOS. Press.
- [24] Lassila, O.; Swick, R. R.; Wide, W.; and Consortium, W. 1998. Resource description framework (rdf) model and syntax specification.
- [25] Protégé, free, open source ontology editor and knowledge-base framework.
<http://protege.stanford.edu/>
- [26] Protege 4.x OWL, a guide,
<http://protegewiki.stanford.edu/wiki/Protege4GettingStarted>
- [27] Pizzas in 10 Minutes, An Ontology for pizza,
<http://protegewiki.stanford.edu/wiki/Protege4Pizzas10Minutes>

[28] Jose Hiram Soltren, Query-Based Database Policy Assurance Using Semantic Web Technologies, Sept 2009.