A Model of Distributed Rights Allocation in Online Social Interaction

A thesis presented in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Information Technology

at

Massey University,

Albany, Auckland, New Zealand

Adnan Ahmad

2013

This research was sponsored by the National Science Foundation (NSF), U.S. under contract no. 0968445. However, the views and conclusions contained in this dissertation are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of NSF, or of the U.S. government or any other entity.

Abstract

In computing, the management of information resources is done through access control, a process by which authorized users are granted permission over resources. The last decade has witnessed the emergence of socio-technical systems (STS) like Facebook, Twitter, and YouTube, where millions of users interact with each other and share billions of resources on daily basis. Access control for a STS is different from traditional systems in having to satisfy the social requirements of the community as well as the technical requirements of the system.

The problems traditional access control models face today are firstly the complexity of mapping millions of users to billions of resources, and secondly the social requirements of users who want to own the resources they post. Current access control models for STS manage access through rule semantics, roles, trust, history management or contents. However, there is no general logical scheme that allows users to allocate rights, covering not just transfer and delegation but also joint and several ownership.

The trend from centralized to distributed access control demands a general model to manage rights allocation for users having heterogeneous privacy policies. The model's validity derives from socio-technical design, where social requirements like ownership, freedom and privacy give technical access axioms. The aim is to satisfy not only technical but also social requirements, over which the success of today's software depends.

This research first proposes the social access control model for supporting local administration, dynamic asymmetric relationships and object privacy classification. This core model is then used as a basis of various rights allocation models. The research further illustrates a rights allocation framework based on various properties of STS and presents a reduction approach to design the model. This framework reduces all the possible rights allocations into four basic models: Replace, Revoke, Share and Merge, which can manage every tweet, every post, and every single communication on any STS. The proposed rights allocation models are demonstrated on various current and hypothetical use-cases of current STS to show that it can be used in any system that has social interactions, and where users want to control their resources. This research extends the online social interactions in STS to new horizons which are currently restricted due to the limitations posed by current technology.

Acknowledgement

My success(es) are not through my individual effort alone, but rather through the combined efforts of many.

~Maori proverb

First and foremost, I would like to praise and thank Allah Almighty for bestowing upon me the wisdom, courage, patience and blessings to complete such a huge research endeavor. Thanks Allah, You have given me the power to believe in my passion and pursue my dreams. I could never have done this without the faith I have in you.

The most important person I would like to thank is Dr. Brian, my supervisor, for his support and care during this long and tiring journey. As a "mathematician is a machine for converting coffee into theorems", Dr. Brian is a mind for converting complex, detailed theorems into amazingly simple '*two-liners'*. Whenever I struck in some problem, he always had a cheerful smile with a mind boggling direction, and a friendly suggestion to enjoy the beautiful weather and God's other blessings bestowed upon us. I have learned a lot from him in terms of research and leading a joyful life – to disconnect from the 'connected' world. He really taught me to appreciate the beauty of life along with my goals. Dr. Brian, I will surely miss our long thoughtful discussions. I wish I could stay with you a little longer as I still have to learn a lot from you.

I take immense pleasure in thanking Dr. Lech, my co-supervisor, for his friendly attitude that he has maintained towards me throughout these years. I found a thoughtful supervisor, a caring elder and a cherish friend in his form. His concern for his every single student, cycle stealing from his busy schedule to adjust one of us was quite impressive. Dr. Lech, you are more than a supervisor to me and for that I am really thankful to you. I wish, I can carry on our relation in the future, and it may get stronger with time.

I would like to thank Dr. Yasir and Dr. Sohaib for their support throughout these years. Their room was the 'chamber of secrets' for our lively discussions, absurd ideas and what not! I especially thank Dr. Sohaib for reading out this dissertation and helping to improve its presentation.

Also, I wish to express my sincere gratitude to Dr. Tony Norris, Dr. David Parsons, Dr. Andre Barczak, Dr. Andrew Colarik, Dr. Daniel Playne and Mrs. Siew Whitworth for their valuable suggestions. I am also obliged to many anonymous conference and journal reviewers who helped to improve the quality of my papers and thus this dissertation quite substantially.

Finally, I am forever indebted to my parents. I can barely find words to express their endless love, unconditional support and encouragement when it was most needed. Mama and Papa, I cannot thank you enough for being there. If it was not for you I cannot accomplish my greatly desired dream!

Table of Contents

List of Figures

List of Tables

List of Publications

Whitworth, B., & Ahmad, A. (2013). The Social Design of Technical Systems: Building technologies for communities. Aarhus, Denmark: The Interaction Design Foundation.

Ahmad, A., Whitworth, B., & Janczewski, L. (2012). A Framework of Rights Allocation in Online Social Networks. *International Conference on Advances in Information Technology.* Bangkok, Thailand.

Ahmad, A., Whitworth, B., & Janczewski, L. (2012). Dynamic Rights Reallocation in Social Networks. *International Information Security and Privacy Conference.* Heraklion, Crete, Greece.

Ahmad, A., Whitworth, B., & Janczewski, L. (2012). More Choices, More Control: Extending Access Control by Meta-Rights Reallocation. *IEEE International Conference on Trust, Security and Privacy in Computing and Communication.* Liverpool, United Kingdom.

Whitworth, B., & Ahmad, A. (2012). Socio-Technical System Design. In M. Soegaard, & R. Dam (Eds.), *Encyclopedia of Human-Computer Interaction.* Aarhus, Denmark: The Interaction-Design.org Foundation.

Whitworth, B., Janczewski, L., & Ahmad, A. (2012). A Logic of Creation in Online Social Networks. Las Vegas, Nevada, USA.

Ahmad, A., & Whitworth, B. (2012). Ethical Issues in Online Social Networks. *International Conference on Networks and Information.* Bangkok, Thailand.

Ahmad, A., & Whitworth, B. (2012). A Reduction Tree for Social Networks. *8th Annual IIMS Conference.* Auckland, New Zealand.

Ahmad, A., & Whitworth, B. (2012). Future Directions in Access Control for Online Social Networks. *International Conference on Networks and Information.* Bangkok, Thailand.

Ahmad, A., & Whitworth, B. (2012). Characteristics of an Access Control Model for Online Social Networks. *New Zealand Information Science Doctoral Conference.* Hamilton, New Zealand.

Ahmad, A., & Whitworth, B. (2011). Access Control Taxonomy for Social Networks. *International Conference of Information Assurance and Security.* Malacca, Malaysia.

Ahmad, A., & Whitworth, B. (2011). Addressing Technical Complexity in Social Networks. *7th Annual IIMS Conference.* Auckland, New Zealand.

Ahmad, A., & Whitworth, B. (2011). Distributed Access Control for Social Networks. *International Conference of Information Assurance and Security.* Malacca, Malaysia.

Ahmad, A., & Whitworth, B. (2011). Towards an Access Control Model for Social Networks. *New Zealand Information Science Doctoral Conference.* Wellington, New Zealand.

Terms used in formulae

Admin	–	Administrator of a system
AC	–	Attestation Certificate
AU	–	Active Users
Ben	–	Beneficiary
D/C	–	Don't Care condition
Dge	–	Delegatee
Dgr	–	Delegator
E	–	Entity
GP	–	General Public
JBen	–	Joint Beneficiary
JO	–	Joint Owner
LR	–	Local Role
MR	–	Meta-Right
Mrg	–	Merge
NS	–	Namespace
O	–	Object
OC	–	Object Class
Opr	–	Operation
PO	–	Primary Owner
R	–	Right
Rep	–	Replace
Rev	–	Revoke
S	–	Subject
SH	–	Stakeholder
Shr	–	Share
SO	–	Secondary Owner
VU	–	Virtual User
UR	–	Use-Right

Chapter 1

Introduction

"The best way to predict the future is to invent it"
(~Alan Curtis Kay)

Authorization is a concept used since people has had the need to protect their valuable assets, and this has been done typically by locks and keys to limit individual access to resources (Ferraiolo, Kuhn, & Chandramouli, 2003). In information technology, authorization is termed access control – a process to grant certain privileges over information and resources to identified users (Sandhu & Samarati, 1994), and it is part of security and privacy in information systems. As modern web systems are mainly about accessing resources, every such request must pass through an access control system. This makes it a key factor in the success of any information system used by individuals or communities.

In order to provide access to appropriate users only, access control systems incorporate the requirements of the application to fulfill them in the most suitable manner (Sandhu & Samarati, 1994). As every application is different, various customized solutions have been proposed based on the application's domain. Some examples of incorporating application requirements in access control are the systems for medical data networks (Morchon & Wehrle,

2010), supply chain databases (Kerschbaum, 2010), peer-to-peer file sharing (Bram, 2003), grid environment (Thompson, Johnston, Mudumbai, Hoo, Jackson, & Essiari, 1999), webOS (Belani, Vahdat, Anderson, & Dahlin, 1998), mobile ad-hoc networks (Kraft & Schafer, 2004), and coalition environments (Freudenthal, Pesin, Port, Keenan, & Karamcheti, 2002). The understanding of application requirements plays an important role in designing the access control system just like understanding the requirements of software is important in directing its design and development (Goguen & Linde, 1993).

The recent past has observed the rapid emergence of socio-technical systems (STS) like Facebook, YouTube, Wikipedia, where information technology mediates a community of users interacting as equal partners (Hippel, 2005). These systems support different types of online social interactions among users and provide platforms where millions of users interact with each other and share billions of resources (Oltsik, 2009; Preece & Shneiderman, 2009). These systems are built around the social requirements of the communities and create virtual online communities similar in structure and requirements as the physical ones (Jahnke, 2009). In this regard, their access control systems not only deal with technical requirements of the systems but also with the social requirements of the communities. If these requirements are not fulfilled by the system, the community will leave and the system would collapse even having the best software design (McInnerney & Roberts, 2004). The access control models for STS differ from traditional models as they involve ownership, freedom, relationships and local control rather than centralized security administration. For example, social networks have introduced local visibility of objects, that is, mapping the resources only to visible users, which significantly reduces the complexity overhead posed by traditional models[1].

The increasing popularity of STS has given rise to new types of security and privacy concerns (Simpson, 2008). Access control in these systems is critical, as a single mishandling of access allocation can cause outrage. For example, it is well-documented that some online social interactions have cost people their jobs when their employer discovered that they had said inappropriate things about their company online (Simonetti, 2004). Sexual predators use social networks to find victims (Poulsen, 2006). Individuals have been stalked due to their personal information being placed on their online profiles (Rowse, 2006). Universities have taken

[1] For Facebook 955 Million users with 67 billion resources, the traditional access control matrix has 246 trillion

disciplinary actions against students posting photographs (Barnes, 2006). In 2010, a teenager was murdered in Australia due to her online activities, and the local police department claimed to have received at least one complain every month in relation to online offenders (Breen, 2010). Currently many STS provide only the basic privacy access controls: a resource can be either completely private or completely public, and so they are struggling to deal with the new demands of online social interaction (Carminati, Ferrari, Heatherly, Kantarcioglu, & Thuraisingham, 2009, 2011).

Initially, the access control models for STS provide ownership and local control. With further evolution, other social requirements are incorporated in access control design to support owner trust (Pujol, Sanguesa, & Delgado, 2002; X. Zhang & Q. Zhang, 2005; Ali, Villegas, & Maheswaran, 2007), rule semantics (Elahi, Chowdhury, & Noll, 2008; Carminati, Ferrari, & Perego, 2009), relationship based roles (J. Li, Tang, Mao, Lai, & Zhu, 2009; Tapiador, Carrera, & Salvachua, 2011), history based decisions (Fong, Anwar, & Zhao, 2009) and content awareness (Hart, Johnson, & Stent, 2007). These models significantly contributed to the area of access control research for STS and are used widely in industry in order to facilitate users. However, none of the current access control models for STS provide the formal semantics of rights allocation – methods in which rights can be assigned to different users, which include rights transfer, delegation, sharing and merge.

Rights allocation is important as it allows the users to collaborate, introduces the opportunities associated with multiple-ownership, and provides a facility to manage meta-rights more efficiently. It is useful in many situations, such as backup of role, collaboration of work and decentralization of authority (Park & Lee, 2005; X. Zhang, Oh, & Sandhu, 2003). As a STS is built around the social requirements of a community, there is need of an access control model that incorporates various types of rights allocation.

1.1 Problem statement

This research was motivated by the absence of a general rights allocation model for STS. Rights delegation and rights transfer are not supported in current STS. Also, the 'merging' of rights of two users and the opportunities associated with multiple-ownership have not yet been explored for these systems. Besides, only rights sharing has been explored for STS and

considered as one of the major reasons for their great success, for example sharing of view rights over Wikipedia articles, YouTube videos, and Facebook photos. But the current rules of rights sharing in STS are based on designer intuitions rather than formal models, so they vary between systems and over time, with public outrage the only check. There is no agreed scheme for allocating permissions to create, edit, delete or view object entities, let alone manage roles.

Moreover, traditional rights allocation models are less useful for STS for the following reasons:

1.1.1 Complexity

The number of users and objects is quite high in STS as compared to traditional systems, (for example Facebook reports 955 million active user accounts each adding many hundreds of photos and comments each year, with more than 100 billion friend connections[2]). So models that map resources to all the users in the system become over-complex for millions of users and billions of resources. Role based access control (RBAC) (Ferraiolo & Kuhn, 1992; Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001; Sandhu, Coyne, Feinstein, & Youman, 1996) is often proposed as the solution to reduce the complexity of traditional systems by introducing roles. However, if traditional access control is about who can enter a system, it grows linearly with number of users, but if STS access control is about who can connect to whom, the relationship combinations – potential access permissions, increase geometrically as a function of the number of users. This global visibility of traditional access control models introduces the challenge of managing the access rights of so many users' contributions, and affects access control efficiency. Also, the frequent content updates and volatile nature of friendship makes it even more difficult to use traditional access control systems for online social interactions (Hart et al., 2007).

1.1.2 Privacy

Connecting to others satisfies relationship needs but also raises privacy concerns (Simpson, 2008). As users contribute online contents, like family or friend photos, they naturally expect to control them. In traditional access control, each user is allocated the same access control policy values, so variants must be requested from a central authority (Hart et al., 2007). Roles in traditional access control systems are system wide groups whose membership is set by a

[2] Facebook statistics page, http://www.facebook.com/press/info.php?statistics, accessed on 30th July 2012.

system administrator. The access rights over user's resource are allocated to a role the user has no control over. In contrast, users on STS want to set their own values without reference to a central authority, for example to share their data with everyone or to restrict it to family and friends. Traditional access control systems do not provide the functionality needed for online social interactions and so struggle with the demand of diverse privacy requirements of today's user (Carminati et al., 2011; Gollu, Saroiu, & Wolman, 2007; J. Li et al., 2009). Generic roles tend to reveal more than the users want, as users cannot specify local requirements using generic roles.

1.1.3 Operations

The set of operations offered by STS is much larger than the set offered by traditional systems. To a traditional access control system, a file can only be read, written to or executed, but users involved in online social interactions want to exercise a much richer set of operations in parallel to that used in physical societies. This sophisticated set not only includes read, write, like/unlike, tag, subscribe and so on, but also other operations to manipulate different objects with different types of collaborations among different users, such as joint ownership of a couple common online persona, transfer of a colleague's research paper to him, or delegation of tracks to track chairs from conference organizers.

1.1.4 Local ownership

STS are built around the social concept of ownership[3] which is not widely supported by traditional access control models. From the set of traditional models, only Discretionary Access Control (DAC) can work with the social structure of STS, but it faces the problems of global visibility of objects, central administration of groups[4], and users' interactional complexity. Another alternate to the complexity problem is RBAC, but it has proved to be even more expensive while supporting ownership (Sandhu & Munawer, 1998a). Besides, traditional access control models assume single ownership of objects, whether it is a person or an organization, but do not support multiple-ownership of objects which may restrict collaboration opportunities in STS.

[3] Either the owner of the content as in the case of YouTube, or owner of the space as in the case of Wikipedia.

[4] Users give rights to groups whose membership is not in their control.

Besides, only rights delegation is explored in reasonable depth for traditional access control models, but other types of allocations like rights transfer, rights share and rights merge are hardly explored in much detail. Also, the delegation in literature is mainly about role to role delegation as permissions are associated with roles. However, STS require domain based delegation where permissions are associated with objects. Various access control models for STS have been proposed in recent years but they lack a systematic scheme for managing rights allocation – methods in which the owner can assign rights to other users, and so lack the opportunities associated with multiple-ownership, rights transfer, rights delegation and rights merge.

1.2 Research objectives and scope

The objective of this research is to design a rights allocation model for online social interactions, which is decentralized, socially valid, and enhances group interactions. The model will support ownership, relationships and local control, and act as a standard for the implementation of rights allocation in any STS, in any programming language over any platform. It may cover future options not yet coded, and take online social interactions to the new level of online group interactions where various geographical/ethnical/religious/political groups interact as a community to users from other groups. It may allow group ownership, joint persona, decentralization of authority, and let users manipulate online objects as they see real world objects.

However, this research only focuses on rights allocation; security is a huge domain. In order to achieve the above mentioned goals, this research assumes some basic considerations that are not included in the scope of this research and neither discussed in this dissertation. Some of those considerations are as follows:

a) It is assumed that an authentication mechanism is in place and all the users are authenticated before they can access system resources. It is also assumed that there is some mechanism to verify the identity of the requestor and so no threat of any identity theft exists. Further, this research does not focus on operating systems or security basics, or mechanisms that come from a blend of well-known techniques.

b) This research does not focus on set of operations over information objects. The model addresses how operational rights are allocated, so is independent of the specific type of objects and operations. However, common operations like create, view, edit and delete are used for explanation.
c) This research does not cover trust mechanism between users, reputation of user in a community or various security leakages and attacks that can happen in online social interactions.
d) This research does not deal with the design principles of STS as software, nor with the load management or credential management systems/architectures of these applications.

Hence this model does not cover the above mentioned functions, that is, authentication, operation types, trust, reputation, software design, and load management. This modularity means that an access control system based on this model can easily be inserted into any current system without modifying other semantics.

1.3 Guide to this dissertation

The rest of the dissertation is organized as follows. The introduction chapter gives a brief overview of the background knowledge, problem domain and research objectives.

Chapter 2 outlines the literature related to traditional access control models, their previous evolution and the challenges posed by the new generation of STS, where communities of people socially interact. Following this basic description of access control models, it further highlights the emergence of STS and the issues related with them. The chapter then explores five different categories of access control models for social networks and briefly reviews the published work in each of them. It then discusses some traditional rights allocation models on delegation, transfer, share and merge, followed by the reasons why they cannot be mapped onto the new requirements of online social interactions. These differences highlight the research gap and lead towards formulating the research question.

Chapter 3 describes the conceptual framework to meet the issues raised in the previous chapter and basic theories related to the design of access control models. It also details the social validity principles used. Further, it presents a rights allocation framework based on the

characteristics of rights allocation and provides a reduction approach to design the models for online social interactions. The proposed framework extends the availability of rights and is used as a basis for different rights allocation models in online social interactions.

After brief overview of the domain requirements, chapter 4 illustrates the social access control (SAC) model, based on ownership domains, local administration, local roles and object classes proposed in general terms. These core components and their interactions merged into an access control model, to give owner control over resources and relationships. The model acts as the core access control model for STS and as the supporting base for various rights allocation models.

Chapter 5 and 6 specify in detail the allocation models in use-rights and meta-rights allocations respectively. The four basic models Replace, Share, Merge and Revoke are explored for use-rights and meta-rights to result in designing eight different rights allocation models. Each model starts with the definition of the operation in the context of particular right and then takes some STS scenario to emphasize the importance and need of the particular model. It further outlines the characteristics used for the reduction approach generation in chapter 3, and uses it to generate the model. The rights of various roles associated with the owner domain are discussed and logical definitions of the models are given, followed by its revocation.

Chapter 7 critically analyzes all the use-rights and meta-rights allocation models. The models' permutations are calculated to estimate all their possible outcomes, along with some generalized rules. These permutations give insights about the behavior of each model and help to generalize some notions and rights equations for both types of rights. It also illustrates the completeness of the models by combining all the characteristics provided in chapter 3, and showing that the proposed models cover all the design options.

Chapter 8 validates the proposed models by demonstrating basic and advanced use-cases from current STS to emphasize that the models are generic enough to provide functionality to most types of online social interactions.

Chapter 9 concludes the dissertation on how research questions were addressed. Subsequently, the implications of the presented research are discussed followed by some limitations and future research directions. The guidelines to this dissertation are given in figure 1.1.

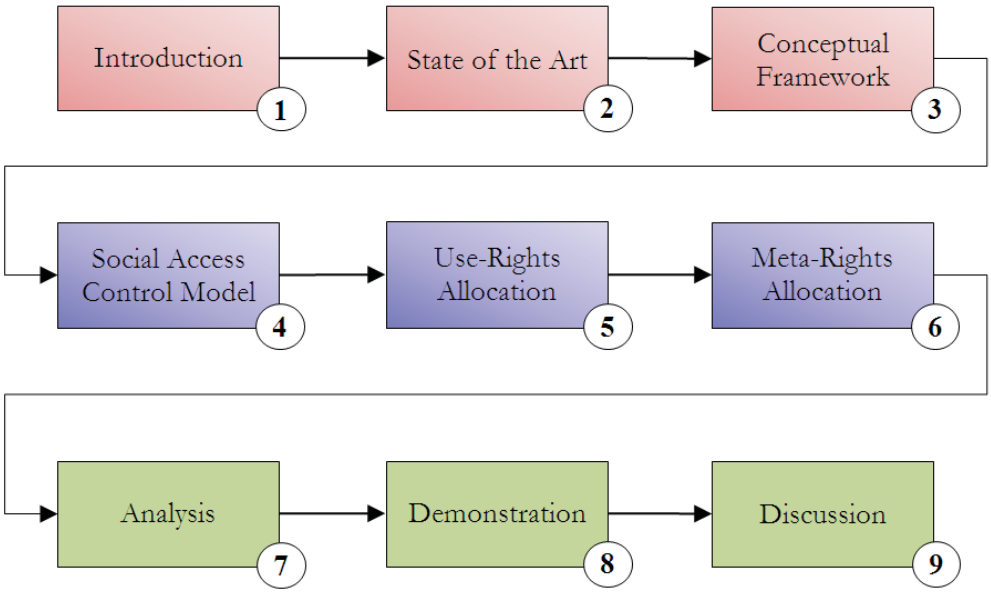

Figure 1.1: Guide to this dissertation

Chapter 2

State of the Art

This chapter reviews the state of the art of rights allocation in online social interactions. The literature for this chapter is gathered based on the three core themes for this research, that is, access control, rights allocation and socio-technical systems. The research in these areas is explored based on 'keyword search' on various research databases[5]. The work is then filtered based on the reputation of the authors, the reputation of the conference/journal and the relevance to the current research.

The chapter starts with the evolution of access control and outlines various popular access control models present in the literature. It highlights the fact that those models were proposed to fulfill the requirements of different applications. The chapter then introduces the emergence of STS along with its various new properties and requirements, which demands the development of different access control models. The chapter further explores various rights allocation models for traditional systems followed by the reasons why they are less useful for STS. This leads to the identification of research problems and formulization of the research question for this research.

[5] Such as IEEE, ACM digital library, Springer, Google Scholar and home pages of various well-known researchers of the field.

2.1 The origin of access control

In computing, rights over resources are managed by access control system, which defines who can do what under what circumstances (Ferraiolo et al., 2003). The need of first access control system arose with the emergence of multiuser computer systems, when people realized the need to prevent one user from interfering the work of others sharing the same system. So, they developed a model that associates users with identities and assigns permissions over system resources based on those identities (Karp, Haury, & Davis, 2010). That earliest model in 1969, introduced the formal notions of subjects and objects, and an access control matrix to hold the access permission of subjects over objects (Lampson, 1969).

An important requirement of information systems is to protect data and resources against unauthorized disclosure (secrecy) and improper modifications (integrity), while at the same time ensuring their availability to legitimate users (availability) (Ferraiolo et al., 2003). Hence, enforcing protection requires that every access to the system and its resources should be controlled, so that only authorized accesses can take place (Samarati & Vimercati, 2001). Access control is arguably the most fundamental and the most pervasive security mechanism in use today. It shows up in virtually all systems and imposes great architectural and administrative challenges at all levels of enterprise computing. From a business perspective, access control has the potential to promote the optimal sharing and exchange of resources, but it also has the potential to frustrate users, impose large administrative costs, and cause the unauthorized disclosure or corruption of valuable information (Ferraiolo et al., 2003). Although access control may sometimes seem conceptually straightforward, it is both complex and error-prone in practice (Abadi, 2009).

2.2 Policies, mechanisms and models

When planning an access control system, three abstractions should be considered. They are access control policies, access control models, and access control mechanisms (Ferraiolo et al., 2003).

2.2.1 Access control policies

Access control policies are high-level requirements that specify how access is managed and who may access what information, and according to which access control must be regulated (Samarati & Vimercati, 2001). While access control policies can be application-specific and thus taken into consideration by the application vendor, policies are just as likely to pertain to user actions within the context of an organizational unit or across organizational boundaries. For instance, policies may pertain to resource usage within or across organizational units or may be based on need-to-know, competence, authority, obligation, or conflict-of-interest factors. Such policies may span multiple computing platforms and applications.

2.2.2 Access control mechanism

An access control mechanism defines the low level (software and hardware) functions that implement the controls imposed by the policy and formally stated in the model. An access control list is a well-known example of an access control mechanism. At a high level, access control policies are enforced through a mechanism that translates a user's access request, often in terms of a structure that a system provides. There are a wide variety of structures; for example, a simple table lookup can be performed to grant or deny access. Although no well-accepted standard yet exists for determining their policy support, some access control mechanisms are direct implementations of formal access control policy concepts (Ferraiolo et al., 2003).

2.2.3 Access control model

An access control model provides a formal representation of the access control policy and its working. The formalization allows the proof of properties provided by the access control system being designed (Samarati & Vimercati, 2001). Rather than attempting to evaluate and analyze access control systems exclusively at the mechanism level, security models are usually written to describe the security properties of an access control system. Access control models bridge the wide gap in abstraction between policy and mechanism. Access control mechanisms can be designed to adhere to the properties of the model. Users see an access control model as an unambiguous and precise expression of requirements. Vendors and system developers see access control models as design and implementation requirements. On one extreme, an access control model may be rigid in its implementation of a single policy. On the other extreme, a

security model may allow for the expression and enforcement of a wide variety of policies and policy classes (Ferraiolo et al., 2003).

2.3 The evolution of access control

The understanding of requirements of the application plays an important role in designing the access control model, in the same way as understanding the requirements of software directs the design and development of it (Goguen & Linde, 1993). In order to allow authorized access only, access control model generally incorporates the requirements of the application (Sandhu & Samarati, 1994). As new applications are developed on users' requirements, new access control models are designed to provide them security and privacy in a consistent manner.

Lampson (1969) introduced a formal model to manage the users, resources and their access. With time, some other models were designed to solve the access control for earlier systems. However, those earliest access control models support centralized monolithic administration, but faced problems with distributed systems (Karp et al., 2010). The emergence of roles in organizations (Ferraiolo & Kuhn, 1992), business domains (Brewer & Nash, 1989), and interactions of multi-domain systems (Freudenthal et al., 2002) has shifted the application domain towards more distributed control. Following is a brief overview of the major access control models:

2.3.1 Bell-LaPadula model

Bell and LaPadula (1973) formulated the military rules for military security applications into a mathematical model. As military security form a hierarchy and higher rank documents are only accessible to higher rank officials, the model introduced the multilevel secure system. Users are only allowed to access information which is classified as lower than their own security clearance. This way confidential information is restricted only to the higher ranked officials.

2.3.2 Biba's integrity model

Bell-LaPadula model was designed for the confidentiality[6] of the data, but does nothing to prevent unauthorized modification of information (integrity). To overcome this drawback,

[6] Unauthorized read.

Biba integrity model (Biba, 1977) was introduced. It was not an alternative to Bell-LaPadula but can act as an adjunct to it. The Biba model allows a subject to read an object, if the object has greater security level than the subject. The model further extends that a subject can only write to an object if the security level of the subject is higher than the object. In general, an object can only be written from the higher levels and read from the lower levels.

2.3.3 DoD models

In 1985, the United States Department of Defense (DoD) published its own standards for military and personal applications, commonly known as MAC (mandatory access control) and DAC (discretionary access control) (TCSEC, 1985). The working domain of MAC was also military applications just like Bell-LaPadula model, so it has implemented the same multilevel security and classified users into multiple security levels. The system has one administrator who controls every system resource and manages its access for all the users in the system. This centralized administration not only suits the military applications but also works for various commercial applications (Jajodia & Sandhu, 1991; Qian & Lunt 1996; Sandhu & Chen, 1998; Morchon & Wehrle, 2010).

As opposed to MAC, discretionary access control (DAC) was introduced to support ownership, local control and other requirements of personal applications. DAC was developed for personal applications and data, and held the owner responsible for the security of their data. The model was based on the Locke's idea of ownership (Locke, 1963) and successfully worked for various applications till date (Belani et al., 1998; Bram, 2003; Thompson et al., 1999; kerschbaum, 2010; Freudenthal et al., 2002).

2.3.4 Clark Wilson model

By involvement of information technology in business, most commercial firms recognized that DAC and MAC were not sufficient for their needs (Ferraiolo et al., 2003; Karp et al., 2010). The commercial users' need was to secure data from unauthorized modifications instead of its secrecy. Also, business processes require some mechanism to protect their clients from untrusted employees. So, Clark and Wilson (1987) formulated a model to satisfy these needs. There are two central concepts in their model, well-formed transaction and separation of duty

(SoD). The former constraints the user to modify data only in authorized way, and the latter ensures that every critical operation must be completed by at least two users.

2.3.5 Chinese wall policy

To facilitate business organizations and understanding the requirements of third party employers and brokers, Brewer and Nash (1989) proposed the Chinese wall security policy. The model distributes the objects in company wise dataset, and further categorized them into conflict of interest (COI) circles. A subject can read an object if the object belongs to the same dataset from which the subject has previously read, and if the subject had not read some other object from the same conflict of interest circle.

2.3.6 Role based access control

In 1992, a study was initiated to gather the requirements of commercial and government organizations, which found that their needs were not being met by access control models of that time (Ferraiolo, Gilbert, & Lynch, 1993). In traditional computer based applications, access control was managed for a known population of predefined users over some known resources through some centralized mechanism (Kane, 2006). In those systems, the number of users and system resources were limited, so the access control models could map every system resource directly to every system user. As the number of users grew, the administrative overhead of managing the users became unsustainable, as well as the system complexity (Karp et al., 2010). The study also explored the fact that in any organization the permissions of some users are similar to one another. The study was followed by a solution to meet these needs, integrating features of existing applications into a generalized Role based access control model – RBAC (Ferraiolo & Kuhn, 1992; Ferraiolo et al., 2001; Sandhu et al., 1996). RBAC introduced system wide roles, which were assigned permissions over resources. In order to access a resource, the user needs to be a part of some role and that role must have access rights over the requested resource. The introduction of roles solved many problems of commercial and government organizations and RBAC became one of the popular access control model (Ferraiolo et al., 2003).

2.3.7 Rule based access control

With further evolution in applications, it was found to be important for an access control decision to support the context. For example, in banking system, the access decisions at day should be different than the same decisions at night, or in military applications, some decision at war may be completely different than the same decision during peace times. To handle such cases, Rule base access control was introduced to support the context in access control decisions (Brucker & Petritsch, 2009; Kirkpatrick & Bertino, 2010; Kulkarni & Tripathi, 2008).

2.3.8 Distributed environments

The challenges with centralized access control models became apparent when the software systems were extended to cross domains (Karp et al., 2010). The agreement of all the companies over rights associated to some role existing in multiple domains proved to be quite difficult. Also, the working rights of one role of one company are restricted to its domain only, which makes the situation more complex. The solution to these problems is given by distributed role based access control (dRBAC) by maintaining separate system domains for each company in the collation (Freudenthal et al., 2002).

2.3.9 Customized access control models

Many other customized access control models were proposed to fulfill the requirements of different applications. This happened because the goal of every application is different and a customized access control model can give the best solution to their needs. Some of the examples of modeling the application requirements in access control are the systems for medical data networks (Morchon & Wehrle, 2010), supply chain databases (Kerschbaum, 2010), peer-to-peer file sharing (Bram, 2003), grid environment (Thompson et al., 1999), web Operating System (Belani et al., 1998), mobile ad-hoc networks (Kraft & Schafer, 2004), and coalition environments (Freudenthal et al., 2002). Table 2.1 shows the major access control models and their application requirements.

Access control was started from earlier systems supporting centralized administration. With evolution, access control logic was developed to offer roles for large organizations and domain based access control for distributed systems. With variations, the traditional access control approach has worked for military and commercial applications, organizational structures,

contextual decisions, distributed applications, medical data, peer-to-peer networks and the grid environment. However, the last decade has seen extreme multi-user system emerge, that is, socio-technical systems (for example email, chat, bulletin boards, blogs, Wikipedia, E-Bay, Twitter, Facebook and YouTube) where millions of users interact with each other and share billions of resources. The permission matrix for social interactions increases geometrically, not linearly, with group size, so the possible connections are astronomical. These systems vastly increase access control complexity, as millions of users want all rights to billions of resources, plus rights to allocate those rights.

Application	Requirements	Model
Mainframe computers	Time sharing, Prevent users from interfering	Lampson
Military applications	Users' dearance, Multilevel security	Bell-LaPadula
Military applications	Hierarchy of users, confidentiality, secure system state, Admin control	MAC
Personal applications	Ownership, No administrator, user control	DAC
Commercial applications	Integrity, Separation of duties, well-formed transaction	Clark Wilson
Commercial and government organizations	System wide roles, complexity, user overhead, organizational structures	RBAC
Organizations	Context aware, situation based	Rule based
Collation environment	Security and availability in overlapping domains	dRBAC

Table 2.1: Evolution of access control models based on application requirements

2.4 Socio-technical systems

Traditionally, the word 'community' is linked to a geographic area such as a neighborhood (Wellman & Gulia, 1999) or with collectivities of people, who share a common experience, interest, or conviction; who experience a positive regard for other members; and who contribute to member welfare and collective welfare (Bender 1978; A. Etzioni & O. Etzioni 1999; Knoke, 1986; Putnam, 2000). However, with the development of communication media, people can go further to broaden the community. Telephone, radio, television and the internet connect distant people with one another without face-to-face communication. Communities built on the information technology are social worlds, where groups of people with common interests and practices communicate regularly and in an organized way (Ridings & Gefen, 2004). These communications are done under the guidance of the community standards and rules to meet the social needs (Rothaermel & Sugiyama, 2001). These communities together

with their rules and communication within it can be viewed as social-technical systems. Socio-technical systems (STS) are social systems sitting upon a technical base, with email as a simple example of social communication by technology means (Whitworth, 2009). They allow people to communicate with each other through technology rather than through physical means.

Socio-technical system theory was originally developed by the Tavistock Institute of Human Relations in the 1950s to explain how new technology impacted primary work systems (Trist, Higgin, Murray, & Pollock, 1963). It was arisen in response to the challenge of understanding complex technical systems that are embedded in a human world (Trist, 1981). In general systems theory (Bertalanffy, 1968), systems form when autonomous (self-directing) parts mutually interact to create equally autonomous wholes. Such systems do not reduce entirely to their parts as their creation involves not just those parts but also complex feed-back and feed-forward interactions. Just as a person is a system of autonomous cells, so a society is a system of autonomous citizens. Such holistic systems whether simple cells or complex people can self-organize and self-maintain (Maturana & Varela, 1998). Therefore, STS is not just social and technical systems side-by-side but the whole unit. Hence STS research is not just applying sociological principles to technical effects (Coiera, 2007), but how social and technical aspects integrate into a higher level system with emergent properties.

Also, it is reported in (Whitworth, 2009, p.4), that as system complexity increases higher system views seem to apply. For example, in the 1950s/60s computing was primarily about hardware, while in the 1970's it became about business information processing, and in the 1980s about 'personal computing'. With the 1990s and email computers became a social medium, and in this decade social computing has flourished with chat rooms, bulletin boards, e-markets, social networks, wikis and blogs. Computing reinvented itself each decade or so, from hardware to software, from software to HCI[7], and now from HCI to social computing. To explain this, Grudin reported three IT levels (hardware, software and cognitive) (Grudin, 1990, p. 2) and Kuutti later added an organizational level (Kuutti, 1996, p. 4). These physical, informational, personal and communal levels show hardware, software, HCI and STS as illustrated in figure 2.1.

[7] Human Computer Interaction.

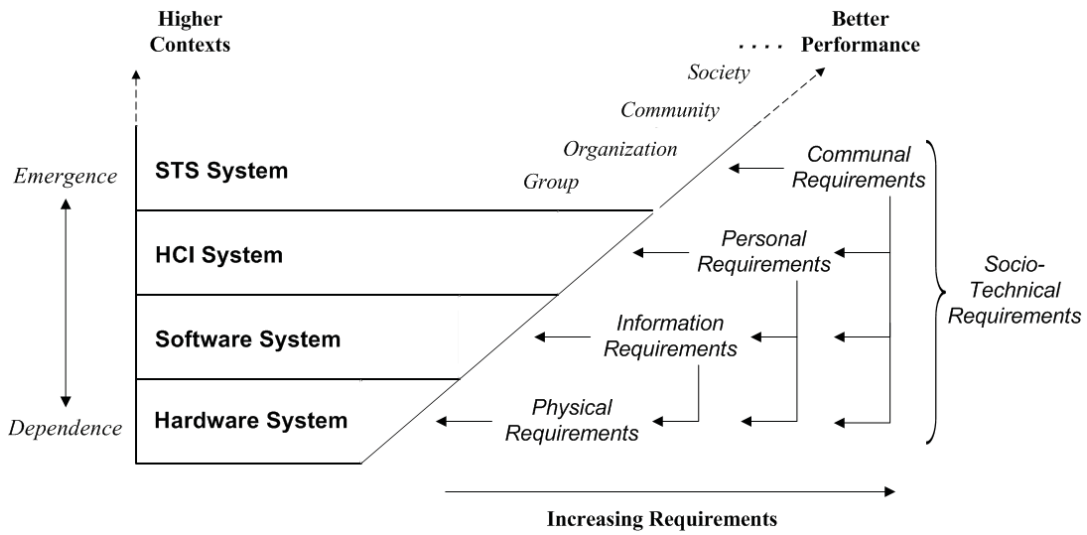

Figure 2.1: Socio-technical system levels and requirements (Source: Whitworth, 2009, page 5)

STS emerge when users utilize information technology to exchange information with other systems in social settings. The implementation of these systems is an ongoing social process as people can form groups, organizations and communities (Kling & Lamb, 1999). A STS is one that involves all four socio-technical levels and their interactions. STS research describes the connections between hardware and software technologies and people and communities. The top of Figure 2.1 is open-ended, as social groups can coalesce into bigger ones, for example, in social evolution people first formed villages, then city-states, then nations, super-nations and perhaps today a global humanity (Diamond, 1999).

The increasing popularity of online communities is giving rise to new classes of security and privacy concerns (Simpson, 2008). The rights management in these social communities is the core thing, as a single mishandling of rights allocation can shift the whole paradigm of the social community. Currently a number of online communities provide only the basic access control: a resource can be either completely private or completely public (Carminati et al., 2009, 2011). The traditional aspect of security is again tested as access control in these online communities is more about access than control. The permission matrix for friend interactions increases geometrically, not linearly, with group size, so for hundreds of millions of users, the possible connections are astronomical. Each account also adds hundreds or thousands of photos/comments a year. Finally, each user wants the same domain control previously reserved only for system administrators. The users allocate access to local roles (Tapiador et

al., 2011) and social structures (Sanders & McCormick, 1993) to restrict a photo to family or friends without asking a central authority. This vastly increases access control complexity as millions of users want all rights to billions of resources.

This research revisits the problems of access control in STS and proposes the rights allocation models in ownership domain by following socio-technical design approach. Even if these virtual societies are new, people have been socializing for thousands of years and the social principles of physical society can work for these online communities as well. The aim is to identify software patterns that embody social principles as well as technical principles. The result would be a consistent scheme to allocate distributed rights in a socially acceptable way that works for any STS.

2.4.1 Socio-technical design

Developing a STS is a system engineering task: not only the software should be taken into account but also hardware, system interactions with its human users and various constraints coming from social policies and regulations (Scacchi, 2004; Sommerville, 2004). System engineering is inherently interdisciplinary involving different engineering disciplines as well as, particularly in case of STS, organizational sciences. STS research explicitly shows the human social dimensions that ought to be taken into account when designing systems involving technology. It recognizes that technology does not exist in a vacuum, but affects those who use it and that they, in turn, affect its design. A STS, then, has a social component and a technical component, and both of these must be integrated and function together smoothly in order for the overall system to achieve its true potential (Davenport, 2009).

If the software system is developed first and the social needs are mapped on the already existed technology, there would always be a gap between the requirements and the final outcome. This is a well-known shortcoming in software engineering that software does not exactly match the requirements of the user. In socio-technical terms, this gap is often termed as socio-technical gap. So as in software engineering, requirement analysis of STS is done first followed by the system development, to fulfill the requirements in the most appropriate way. It will reduce the overhead for the designer (the system design matches the requirements), the developer (know what to build and how) as well as the end user (the product matches the social needs). The challenging problems related to the analysis and design of a STS are the

problems of understanding the requirements of its software component, the ways technology can support human and organizational activities, and the way in which the structure of these activities is influenced by introducing technology (Clegg, 2000; Gregoriades, Shin, & Sutcliffe, 2004).

2.5 Types of STS

STS are built around the social requirements of the community and create a virtual online community, which is similar in structure and requirements as the physical ones (Sproull & Arriaga, 2007). They can be divided into many types based on their purpose and functionality. Some of the common STS types are as follows (Whitworth & Ahmad, 2012, p. 37):

a) **Communicate**: STS like email, chat and instant messages allow users to communicate with each other in a private manner.

b) **Learn**: STS like WebCT and Moodle allow teachers and students to promote distant learning by establishing a virtual classroom environment.

c) **Trade**: STS like E-bay and Amazon allow users to exchange goods with each other in a trusted manner.

d) **Work**: STS like Monster allow users to find and offer work easily.

e) **Friends**: STS like Facebook, Orkut and MySpace allow users to communicate with their friends and family by making a social circle.

f) **Knowledge**: STS like EasyChair and Wikipedia allow users to exchange knowledge with the community through open web encyclopedia and/or reviewed research settings.

g) **Download**: STS like Webdonkey and bit-torrent allow users to download files and software.

h) **Play**: STS like second life allow users to play virtual world games, communicate with other virtual friends and allow an experience which is impossible in reality.

i) **Keeping Current**: STS like Digg and Delicious allow users to remain up-to-date and provide others a chance to look at most viewed topics in the desired area of interest.

j) **Media Sharing**: STS like YouTube and Flickr allow users to exchange media files with the community.

k) **Follow**: STS like Twitter allow users to forms a group view by linking leaders and followers.

l) **Advice**: STS like Internet forums allow users to get advice/information from one another.

Online social interaction is a communication instance between users in these STS. It can be divided into three major types, with respect to the source (sender) and the sink (receiver): one-to-one, one-to-many, and many-to-many.

2.5.1 One-to-one

In STS supporting one-to-one online social interactions (like communicate, play, trade and so on), the owner gives the access rights to the desired user set over a particular resource and deny all the other users in the system. So, it is not necessary to map every resource in the system with all the users, who are not the candidate for any access right over the resource.

2.5.2 One-to-Many

In STS supporting one-to-many online social interactions (like advise, media sharing, learn and so on), the owner gives the access rights to the desired group of users on a particular resource and deny all the other users in the system. Also, sometimes the owner grants write access to one group but read access to the whole community. This requires local roles for every user where they can decide the access for different roles over different resources.

2.5.3 Many-to-Many

In STS supporting many-to-many online social interactions (like knowledge, download, keeping current and so on), a set of users having administrative rights over the resource decides its access for various groups. This requires the support for multiple administrative authorities for different types of resources and roles. (See Table 2.2)

Among all the mentioned types of online social interaction in table 2.2, social networks can be seen as the richest example of supporting all three types. Before going into the details of rights

allocation in online social interactions, the next section outlines some of the current access control models for social networks.

Type of STS		Type of online social interaction		
Purpose	**Example**	**One-to-One**	**One-to-Many**	**Many-to-Many**
Communicate	Email, chat, IM	Yes	Yes	No
Learn	WebCT, Moodle	Yes	Yes	No
Trade	E-bay, Amazon	Yes	Yes	No
Work	Monster	Yes	Yes	No
Friends	Facebook, MySpace	Yes	Yes	Yes
Knowledge	Wikipedia	No	Yes	Yes
Download	Webdonkey, Bit-torrent	No	Yes	Yes
Play	second life	Yes	No	Yes
Keeping current	Digg, Delicious	No	No	Yes
Media Sharing	YouTube, Flickr	No	Yes	No
Follow	Twitter	No	Yes	No
Advice	help boards, AnandTech	No	Yes	No

Table 2.2: Type of STS and their respective online social interactions types

2.6 Current access control models for social networks

Social networks are systems based on relationships and friendships. These systems allow the user to create a self-descriptive profile – which is used to share information about them, to make connections with other users. The social need of relationship building is the underlying requirement of these systems, and they allow their users to establish, develop, and maintain 'social networks' through publishing and linking 'profile' pages created by other users. These systems have three major features: a) They allow the users to add other users as their friends, b) they allow the users to exchange asynchronous email messages and synchronous online chat, this feature is further extended to post messages on one's wall to propagate to all friends, and c) they support user oriented groups, which one can joint based on common interest or experience (boyd, 2006; boyd & Ellison, 2007). During the last decade, these systems have got enormous attention from users which can be seen by the presence of more than 955 million users on Facebook, where they share billions of resources (Carminati, Ferrari, & Perego, 2006).

This section covers various access control models for social networks and their distinguish properties. The access control models for social networks have mainly focused on ownership,

relationships and local visibility and span over the following five categories: a) Trust based, b) Role based, c) Rule based, d) history based and, e) content based. Following are some of the current access control models for social networks:

2.6.1 Trust based access control

The concept of trust has an important role in social exchange theory (Roloff, 1981), and has highly influenced the dynamics of our social and individual interactions. It may be defined as "the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party" (Mayer, Davis, & Schoorman, 1995, p. 4).

The idea of using trust as an indicator of some characteristic is not new. One of the most well-known trusts metric is Google's PageRank (Page, Brin, Motwani, & Winograd, 1998), which is used to evaluate the trustworthiness of websites. Trust mechanism is generally based on the reputation of the involved entities, such as online transaction sites, or third party certificate issuing authorities. In social networks, trust plays its part when a user joins the network and starts making connections with other users to form his/her own trusty network.

As there are some access control models based on trust for Peer-to-Peer file sharing (Tran, Hitchens, Varadharajan & Watters, 2005; Kamvar, Schlosser, & Garcia-Molina, 2003), mobile ad-hoc collaborative environment (Adams & Davis, 2005) and trust for RBAC (Dimmock, Belokosztolszki, Eyers, Bacon, & Moody, 2004), the same approach has been explored for social networks as well. It is proposed that social trust can be used as an access control decision parameter in social networks rather than giving user the control over the resources.

For instance, a multi-level security approach is adopted in (Ali et al., 2007), where access is granted on the security level of the requestor and community trust is the only parameter used to determine the security level of the users and the resources. Each user is assigned a reputation value as the average of the trust rating given by other users in the system. Every resource is assigned a confidence level equal to the trust level of the owner, and only users with equal or higher trust level can access it. It means that the maximum security value of a resource is the maximum trust value assigned to the owner by the community. To enforce this, the resources are encrypted using a threshold based secret sharing scheme (Shamir, 1979). For

each resource, the owner generates a secret key *K*, which can be split into *n* portions and then reconstructed only by using *x* portions of it. The *n* portions of *K* are distributed among trustworthy nodes – based on owner social circle, prestigious nodes or at random among users. When a requestor tries to access some resource, he needs to retrieve the *x* portions of *K* from the set of *n* nodes and then decrypts the challenge for that resource. These portions are only released if the requestor trust level satisfies the resource confidence level.

A theoretical online trust formation model is explored in (X. Zhang & Q. Zhang, 2005), based on the theories of social exchange (Blau, 1964), reasoned action (Fishbein & Ajzen, 1975), planned behavior (Ajzen, 1985, 1991), and expectation-confirmation theory (Oliver, 1980). The authors have distinguished trust in three different approaches, that is, cycle approach, stage development approach and factor approach, and use them to form an integrated model for online trust formation. The cycle approach is about keeping the trust of a user, and is based on the assumption that the satisfactory outcomes of the prior actions positively affect the trust over that party (Deelmann & Loos 2002; Fung & Lee, 1999). The stage development approach deals with trust in different stages and shows that trust starts with initial stage and may transform to committed stage if increased by positive outcomes (Ba, 2001; Kim & Tadisina, 2003; McKnight, Choudhury, & Kacmar, 2000; Shapiro, Sheppard, & Cheraskin, 1992). The factor approach identifies different factors that can affect trust and discusses the weighted multi-dimensional approach towards trust formation (Kim, Song, Braynov, & Rao, 2001; Pavlou, Tan, & Gefen, 2003; Yoon, 2002). An integrated model based on all the three approaches is presented in three layers: the beliefs-attitude-intention-behavior logic is the fundamental layer, the core of the model is common trusting belief, system trust belief and situational decision to trust, and five critical factors from which trust can be formed and enhanced. These factors are trustor factors, trustee factors, system trust factors, interaction factors and external environment factors. In addition to that, the model also illustrates the dynamic two stage development, that is, initial trust and robust trust.

An algorithm similar to Google's PageRank is explored for evaluating links for social networks, and is termed as NodeRank (Pujol et al., 2002). The approach uses a trust metric to calculate the users' reputations, and is based on the links between users. The reputation of a user is calculated based on the analysis of his position in the social network. The algorithm associates a degree of authority with every user, which reflects his/her reputation within his/her

community and is calculated as a function of total authority present in the social networks and the authority of incoming links to the user. The main idea behind NodeRank is that each user has an authority and a part of it is propagated to other users through his/her outer links and so the authority of a user depends on the authority of the users with in-links. This approach has been criticized as trust is multi-dimensional and human psychology evaluates it on various parameters, which cannot be covered by a single connection based trust evaluation (Meo, Nocera, Quattrone, Rosaci, & Ursino, 2009; Kate, 2009).

Some other interesting approaches to evaluate trust among users in social networks are also explored in literature. For example, Kate (2009) and Golbeck (2009) proposed that users' profile information and internet activity should be taken into account while evaluating the trust, because it is how people evaluate trust in the real world. Also, the work to standardize the trust metric and mechanism to use it with current social networks was discussed in (H. Liu et al., 2008; Massa & Avesani, 2007; Matsuo & Yamamoto, 2009). However, the similarities among the interests of two individuals do not indicate any psychological trust between them.

2.6.2 Rule based access control

Rule based access control provides the facility to the owner to implement their own privacy policy based on some pre-defined rules. Models in this category normally define some rules and then users are granted access based on those rules. This class of access control models easily accommodates contextual information and provides the flexibility to introduce rules according to the precise requirements of the application (Brucker & Petritsch, 2009; Kirkpatrick & Bertino, 2010; Kulkarni & Tripathi, 2008).

In the category of rule based access control models for social networks, a semi-decentralized access control model is presented based on relationship types, trust metrics[8] and degree-of-separation policies for sharing information on social networks (Carminati, Ferrari, & Perego, 2006, 2007, 2009). The model allows the owner to specify the access rules and authorized users in terms of relationship type, depth, and trust level existing between the owner and the requestor. So the relationships are also represented as a tuple with actors along with the type,

[8] The models which consider trust just as a rule/parameter are discussed in this section, whereas the models which are primarily based on trust (or some other parameter) are discussed in their respective sections.

depth and trust level. Additionally, each access request is represented by the owner of the resource, the requestor and the system. When a requestor requests some resource, (s)he receives a set of rules from the resource owner regulating the release of the requested resource. These rules state the type of relationship, maximum depth and minimum trust level that must be present between the owner and the requestor. The requestor then has to acquire the proof from a central node showing the desired relationship type, depth and trust level. Access decisions are made locally at client side; however, to avoid the forgery of proof, semi-centralized certificates are used, where a central node is responsible for managing the certificates.

A well-known drawback of the work proposed by Carminati and colleagues (2006, 2007, 2009) is its vulnerability to privacy threats. The access rules contain the type of relationship and stores at a shared repository which may lead to privacy breach if the owner wants to keep certain types of relationship private to themselves. The authors overcome this drawback in (Carminati & Ferrari, 2008) by proposing an alternative way for the enforcement of access control rules. As opposed to the semi-decentralized approach, the access control is enforced through distributed collaborative process started by the owner. Instead of storing the rules at the server side, the owner contacts the nodes that satisfy the access control rules, so avoiding the privacy breaches. This privacy patch is done by modifying the approach of access grant decision, where the owner is in command of the distribution path rather than issuing the rules and lets the requestor takes the charge. To avoid forgery and trust disclosure, digital signatures and encryption techniques have been used.

To extend their previous work (Carminati et al. 2006, 2007, 2009), a more precise access control model for social networks is proposed in (Carminati et al., 2011), where the user information is encoded using ontologies. The work illustrates the modeling of a social network based on users' profiles, resources, relationship among users, relationship between user and resources, and actions. As various relationships and inferences about them can be easily generated in ontologies, the access control decision uses this knowledge along with the security policies stated as rules. A centralized security kernel is used to enforce the privacy policies at the server side.

All the approaches proposed by Carminati and colleagues (2006, 2007, 2009, 2011) have some common deficiencies. For example, the relationship between two users is considered static which is mostly dynamic in real social networks scenarios. Online relationships start by adding each other as acquaintance which matures into friends and sometimes close friends. The same happens to the trust level, which increases by socializing. Storing the static certificates at the beginning of a relationship can adversely affect the purpose and growth of social networks. Also, the centralized management of certificates exposes a single point of failure and may become processing bottleneck.

In another ontology based work for social networks, a distributed Friend of a Friend (D-FOAF) identity management system and access control model was proposed (Kruk, Grzonkowski, Gzella, Woroniecki, & Choi, 2006). The access control model is based on the social structure of users in terms of friendship level existed between them. An access is granted if the requestor meets the trust and friendship level criteria as mentioned by the resource owner. However, the authors only considered single type of relationships which was later extended to multiple relationship types between the two users in the subsequent work (Choi et al., 2006).

Another ontology based semantic web access control model manages the access to the resources on the basis of relationships between the users and the community (Elahi et al., 2008). The model adopts the Web Ontology Language (OWL) approach and instead of defining the exact rules, uses Jess rule inference engine to execute the inferred semantic rules. The implementation has used the centralized architecture, which may not be very scalable keeping in view the size of social networks. This centralized approach also introduces a shared relationship repository which may compromise the individual privacy, if someone wants to keep certain types of relationship private to themselves.

2.6.3 Role based access control

Role based access control (Ferraiolo & Kuhn, 1992; Ferraiolo et al., 2001; Sandhu et al., 1996) is one of the most popular access control model present in the literature. RBAC introduced system wide roles which are assigned permissions over resources. In order to access the resource, the user needs to be a member of some role which has access rights over the

requested resource. Some access control models based on RBAC advantages also have been proposed for social networks. Following is the description of some of these models:

Tie-RBAC was proposed to support users in online communities by introducing roles (Tapiador et al., 2011). The claim of that research was the similarity between social networks and organizations. As the latter has administrators, operators and organizers so former has colleagues, classmates and acquaintance, so the access control in social networks can be seen as an extension of organizational RBAC. A relationship is considered as a tie between the two users, and is equivalent to the assignment of an actor to a role. The model allows actors in the social network to define their own relations and provides a method for actors to concede access rights to their contacts at the time of establishing the relationships. Relationships are non-reciprocal as *Alice* can consider *Bob*[9] as friend but *Bob* can state *Alice* as acquaintance. A centralized server stores access control policies and is responsible for enforcing them.

Another extension of the RBAC model for social networks is explored in (J. Li et al., 2009). In that approach, the system is divided among user set, role set, resource set and permission set. The user set includes all the users in the system, role set is extracted from social relationships between users, resource set denotes all the resources shared by users with every resource identified by a unique identity RID[10], and permission set includes the set of available operations. In their proposal, the resource owner defines his own set of roles and access rights over resources related to those roles. When a requestor requests some resource, the RID and the requested operation is sent to the owner. On receiving the request, the resource owner retrieves the role relation between the two users from the server and grants the access if the requestor exist in some role and the role is assigned the access permission over the requested resource. The access control module follows a semi-centralized approach, where the server is in charge of role relation management, and the client is in charge of resource and permission management.

Both of the approaches discussed in this section (J. Li et al., 2009; Tapiador et al., 2011) only considered static relationships but do not handle the dynamic nature of relationships. Static

[9] The two most convential placeholder names used in computer security. These names are used for convenience and help technical topics to be explained in a more understandable manner.

[10] Resource Identitifier.

modeling of relationship ignores the fact that relationships matures with time and interactions – if two users start their relation as acquaintances it may take some time before they get the status of close friends. Also, both the models use the centralized server, which can be a bottleneck for the efficient implementation of the scheme due to the enormous load in social networks. Additionally, the centralized server approach may expose a single point of failure and some denial of service (DOS) attack or intrusion to the server may result in compromising the whole system. Furthermore, the centralized approach stores the privacy policies of all the users at the same server, which may lead to privacy breaches as the privacy policies of both the owner and the requestor reside on the same repository[11].

2.6.4 History based access control

Another interesting approach to manage access in social networks is history based access control, which takes the history of the requestor into account and grants the access if the history satisfies the desired criterion. The key idea behind this class of access control models is to maintain a history of access requests by all the users, and use it to grant access to trustworthy requestor while rejecting the malicious ones (Edjlali, Acharya, & Chaudhary, 1998, 1999). In a history based access control model for social networks, the history of previous access requests and their outcomes are used as an identifying factor to handle the future access requests between the owner and the requestor.

A history based access control model for Facebook style social networks is presented in (Fong et al., 2009), where authorization is based on two factors, communication history and acquaintance topology. Communication history is the set of past events between the two users, including the friend request invitation from one user to the other and its acceptance. A communication event is any communication primitive that one user initiates and addresses it to the other. The communication event cannot be out of sequence as the initiation and the response are ordered pair with respect to time. The communication state between any two users is local but occasionally consumes global information – communication history of other users. Acquaintance topology is the social graph gathered by the communication history between users of social networks, and provides a global topology for assisting access control decisions. The users of the social network make the graph vertices and the existence of

[11] Where the owner and the requestor both have access to that repository.

communication history defines the edges. Acquaintance topology is introduced to simplify the communication history as evaluating an access control request on the basis of all the communication history is inefficient. Every owner is responsible for managing his/her privacy policy and whether other users have access to his/her resources depends on his/her policy and the requestor's relationship with him/her.

However, the approach does not differentiate among different types of relationships, but only provides coarse social circle which may restrict the resources to completely public or completely private. Also, the work does not support asymmetric relationships to deal with different perspective of users about each other.

2.6.5 Content based access control

Content based filtering is widely used in literature to solve various problems (Amati & Crestani, 1999; Churcharoenkrung, Kim, & Kang, 2005; Kim, Hahn, & Zhang, 2000), however, it has recently got some attention in the social network research. Content based access control decides the outcome of a request based on the contents of the requested object (Bertino et al., 2003; Samarati, Bertino, & Jajodia, 1996). In general, the contents of an object are used to tag it with labels and the access control policy specifies those labels (Bennett, Dumais, & Horvitz, 2002; Collins & Singer, 1999). These schemes only provide partial automatic response and in essence learn from their own mistakes, so they are not employed at full scale but only to reduce the administrative overhead.

One of the examples of content based filtering in social network is the spam filter proposed by Boykin and Roychowdhury (2005), which works on the contents of users' messages by exploiting various characteristics of social networks. The anti-spam tool distinguishes commercial emails and spam messages from legitimate emails. Another interesting work about filtering the user's wall on Facebook based on the contents of the post is done in (Vanetti, Binaghi, Carminati, Carullo, & Ferrari, 2010). The key idea is to apply user-based content filtering on the messages posted on one's wall and filter out the unwanted messages. The approach uses machine learning text categorization procedures (Sebastiani, 2002) to assign a category to every message based on its contents. However, the work on access control for social networks, based on the resource contents, is hardly explored by any research work. This may be due to less-than-100% efficiency of content based filtering approaches.

In addition, a content aware access control model is presented in (Hart et al., 2007). The model decides the access decision based on the contents of user data by tagging all the documents and users, and uses machine learning techniques to extract the meta-information. The model uses existing relationships between users to decide the authorized users, and identify the resources on their contents rather than some resource identifier. The work has introduced user readable policy specification and its automatic application. However, as recognized by the authors, the system makes occasional mistakes and needs some feedback mechanism to correct erroneous tags. Also, the work does not provide any access control enforcement mechanism, and only considered direct relationship between the owner and the requestor.

The literature highlights interesting patterns in current access control models for social networks. The models have used different approaches to solve the access control problem, but they all have common grounds of ownership and relationships. These two concepts are fundamental to the growth of a community whether offline or online. In traditional access control models, ownership is only supported by discretionary access control (DAC) driven models, as mandatory access control (MAC) is centrally administered and role based access control (RBAC) has proved to be quite expensive while implementing ownership (Sandhu & Munawer, 1998a). Owner oriented relationship requires owner based local roles in DAC driven access control model. This was implicit in most of the access control models for social networks discussed in this section and explicitly defined in (Tapiador et al., 2011; J. Li et al., 2009). However, there does not exist any access control model that supports rights allocation for social networks or any other STS.

2.7 Current rights allocation practices

Today, STS cannot prosper without rights allocation. While currently there does not exist any formal rights allocation model for online social interactions, some systems have some functionality that can be considered as advancement in this direction. For example, sharing of rights can be seen as one of the major reasons for glorious success of STS during the last decade. Revocation is also supported as it is necessary to support sharing. However, delegation and transfer of rights is not supported by any STS. Also, rights merge is not supported by any current STS. Moreover, the opportunities associated with multiple-ownership are not yet explored for these systems. This section covers some of these approaches taken from the

examples of current systems, and highlights the absence of rights allocation along with the opportunities associated with them.

2.7.1 Knowledge management services

Knowledge sharing is a behavior when one disseminates their knowledge and experience to others (Fan & Wu, 2011). To leverage the knowledge effectively, people participate in activities of both knowledge contribution and knowledge seeking (Bock & Kankanhalli, 2006). Knowledge management systems are platforms for people to share knowledge, where conference systems, knowledge repositories and wikis come under this category (Kalman, Monge, Fulk, & Heino, 2002; Majchrzak, Wagner, & Yates, 2006).

Social interaction has been viewed as an important trigger for knowledge creation and sharing, where people have different types of relationships with each other. For instance, conference chairs delegate some of their rights and responsibilities to track chairs. Authors share the rights over a paper with their coauthors, and organizing committee allocates and revokes rights for attendees based on current conference sessions. The ability to allocate social rights is the key to meet these social requirements. This allows STS to evolve from an initial state of one administrator with all rights to a community with delegated and shared rights.

However, current systems do not allow delegating tracks and mini-tracks to different chairs in a running conference. The delegation of online responsibilities is not supported while it is commonly practiced in the physical world. Also, papers are joint property of all of its authors[12], but current systems only allow the submitting author to control all the properties associated with it. In order to give greater legitimacy (Whitworth & deMoor, 2002, 2003) for online social interactions, online delegation should be introduced along with other allocation options. The one author paper scenarios should be changed to allowing all the authors having joint control over editing, versioning, and withdrawing of the work. For example, a many-author paper submitted online can let *one* author alone edit it (transfer), let *one* author edit as allowed by the primary author (delegate), let edits proceed only if confirmed by *all* authors (merge), or let *any* author do any edit (share).

[12] As they have contributed to its completion.

2.7.2 Social networks

Social networks are systems based on relationships and friendships. These software systems are built around the social requirements of friendship and allow users to communicate with their friends by creating a social circle (boyd, 2006; boyd & Ellison, 2007).

Current popular social networks like Facebook allow the owner to control the privacy setting of their content, which are by default private[13]. Other users are added to the social circle of the owner and rights over resources are managed across the social circles. Privacy settings can limit the access of resources to friends, friends of friends or to everyone; however there are currently 275 settings on a Facebook profile, far more than most users can keep track of. Despite of the privacy support by current social networks, some concerns are still raised as any friend of yours can tag you on a photo without your consent and it is displayed as a photo of yours. If people create their personas, they should be able to own it[14], but one cannot delete his/her Wikipedia or WordPress profile[15].

2.7.3 Video sharing services

Video sharing services allow users to distribute their videos through public or private channels. There are two major types of video sharing services, that is, paid and free. The paid services only allow the subscribed users to watch a video, while free services (like YouTube) allow everybody to watch stuff. Irrespective of their apparent different types of rights allocation, both the services share the video contents to restricted or unrestricted users in the community. The rights management in paid services restricts the number of views to particular users; while free video sharing services allow everybody to watch a video for any number of times. Irrespective to their similar nature, there is no agreed formal rights allocation model for these video sharing services to operate on desired set of rights.

The current rules of online social interactions are based on designer intuitions rather than formal models, so they vary between systems and over time, with public outrage the only

[13] Previously Orkut did it differently by allowing everybody to look at your messages and profile information.

[14] By freedom, one should own one's online self.

[15] See how to permanently delete your account on popular website here: http://www.smashingmagazine.com/2010/06/11/how-to-permanently-delete-your-account-on-popular-websites/

check. There is no agreed scheme for allocating permissions to create, edit, delete or view object entities, let alone manage roles.

On Facebook, everything is private and owner can share information with level one friends, level two friends or with the whole community, conversely it was public in Orkut as every information was visible to everyone. On YouTube, videos are publically viewable as everyone can watch a posted video, while it is restricted on paid video sharing services. On Wikipedia, articles are publically editable – everyone can modify the contents of a page; on the contrary, conference systems do not allow anybody but the corresponding author to edit the paper. However, rights allocation in tagging a photo on Facebook is similar in nature as having a co-authored paper on Easy Chair. Also, allowing everybody to watch a video on YouTube is not very different from allowing everyone to edit an article on Wikipedia or allowing a restricted audience to watch the same video on a paid video sharing service.

There is a need to define a formal model for online social interactions based on socio-technical design, which may provide a unanimous framework to be used in most of the current applications. The model may not only support the current practices in STS but also extends the rights allocation to support rights delegation, rights transfer, rights sharing, joint rights and multiple-ownership. This will allow the conference chairs to delegate tracks and mini-tracks to different track/mini-track chairs, conference systems to provide different real world options to authors, solve various identified problems in STS, and give an undisputed rights allocation model for all the other cases. The introduction of rights allocation model for online social interactions may open up new research directions in this field. To highlight the significance of the proposed work, the next section focuses on rights allocation work supported by traditional models and emphasizes the fact why they cannot be used for online social interactions.

2.8 Rights allocation for traditional models

Rights allocation refers to the methods that can be used by the owner to allocate rights to other users. Specifically, it is associated with the methods in which the actor entity in the rights triplet can be modified. It covers the set of possible operations on rights such as rights

delegation, rights transfer, rights sharing and rights merge. Table 2.3 shows the list of possible set of operations on rights[16]:

Operation	Term	Definition
Replace Temporary	Delegation	Temporarily gives rights to another user who can exercise them on the owner's behalf, where the previous user cannot exercise it (Gasser & McDermott, 1990).
Replace Permanent	Transfer	Permanently gives rights to another user who becomes the new owner, and the previous user cannot exercise it anymore (Barka, 2002).
Share	Sharing	Gives away rights over an object while keeping them at the same time, so both users can exercise it (Y. Liu, Yu, & Hao, 2009).
Merge	Joint ownership	Merges the rights of the entire user set over an object, so all parties must agree to perform some action (Ilic, Michahelles, & Fleisch, 2007).

Table 2.3: Various operations on rights and their definitions

Current access control models for STS do not support any type of rights allocation, however some other traditional access control models support some types of allocation. This section covers the previous work on rights allocation for traditional access control models and highlights the differences which make it difficult to use the existing access control models for online social interactions.

2.8.1 Delegation

Delegation is a process which allows a user *Alice* to authorize another user *Bob* to access resources on her behalf (Abadi, Burrows, Lampson, & Plotkin, 1993; Gasser & McDermott, 1990; Varadharajan, Allen, & Black, 1991). There are two delegation views: administrative directed delegations and user directed delegations (Linn & Nystrom, 1999). In the former, each delegation request should be approved by some security officer, while the latter allows the users to delegate rights on their own responsibility. However, in both cases, it is important to have some proper delegation model to prevent unwanted flow of the delegated rights.

[16] See section 3.2.6 for details.

The delegation work in traditional access control models can be classified into three categories: a) machine to machine delegation, b) user to machine delegation, and c) user to user delegation (Abadi et al., 1993; Barka & Sandhu, 2004; Gasser & McDermott, 1990; Gladney, 1997). Following are some of the details of these categories:

2.8.1.1 Machine to machine delegation

Machine to machine delegation is a process which allows a computer based process to authorize another process to act on the behalf of the former (Gasser & McDermott, 1990). It mostly deals with scenarios in the context of distributed systems when one local process requests some resource on the remote machine, and delegates rights to the remote process to request the resource on the former's behalf.

The delegation of rights from local process to remote process was explored in (Gasser & McDermott, 1990) with a mechanism for users to explicitly terminate it. The delegation framework deals with single level delegation and chained delegation. Where single level delegation covers the cases when a single process from local machine deals with single process on the remote server; and chained delegation covers the cases when more than one system is involved between the local and the remote server. Additionally, limited delegation is considered and two options were given, that is, one process can delegate the rights over a subset of its authorized resources, or it can delegate a subset of rights over its complete authorized resource set. The work authorized the claimed delegation by public/private key cryptography (Rivest, Shamir, & Adleman, 1978) to verify that the information is securely transferred between processes.

Another example of machine to machine delegation was explored in (Abadi et al., 1993). The work had proposed single level delegation model that operates on a subset of rights. It also provided powerful mechanisms to implement the delegation with and without the use of signed certificates. However, the delegation model is not adequate for users as they cannot refresh delegation certificates so often and longtime certificates introduce a security threat of being abused by users' delegation.

Another work of a similar type used the term proxy to discuss how one object can delegate its rights to another object to act on the former's behalf (Varadharajan et al., 1991). The authors

claimed that process proxy situations arise in object oriented systems where multiple objects cooperate with each other to perform some task. The research allowed multi-step, partial machine to machine delegations for limited durations, and introduced various mechanisms for its revocation. To prevent the forgery and to maintain the system trust, the instantiation was done using two encryption solutions, that is, public key based solution, and secret key based solution. The research also proposed the extension of Kerberos mechanism (Miller, Neuman, Schillcr, & Saltzer, 1987; Steiner, Neuman, & Schiller, 1988) to implement delegation.

2.8.1.2 User to machine delegation

User to machine delegation can be defined as a process which allows users to authorize a process to act on their behalf (Gasser & McDermott, 1990). It is done in every computer application in the context of human-computer interaction, where a process acts on users' behalf. Thus the user has to delegate his/her rights to the process in order to execute some instruction or to fetch some information from another process.

The work of Gasser and McDermott (1990) in machine to machine delegation also explicitly explored the characteristics of user to machine delegation. Instead of only focusing on the delegation between local and remote processes, the authors also considered the delegation of rights form the user to the local process. The delegation model covers complete design considerations and starts when a user signs in to a machine. If then the local machine requires some resource on the remote machine, the user delegates their rights to the local process to request the resource from the remote machine on their behalf. However, if there are multiple remote machines involved in fetching the desired resource, the local process then delegates the rights to the remote process, so it can request the resource on the behalf of the local process. The model ensures that the delegation is authorized and authenticated, and can be revoked when needed. It also allows partial delegations where a process can only exercise limited rights over subset of resources.

2.8.1.3 User to user delegation

User to user delegation is a process which allows a user to authorize another user to act on the former's behalf (Gaaloul, Schaad, Flegel, & Charoy, 2008). This type of delegation is normally exercised in collaboration environments, where one user may not able to perform some task and/or for distribution of authority (Park & Lee, 2005). Most of the research in this area deals

with role delegation in role based access control (RBAC) environment (Barka, 2002; Sandhu, Bhamidipati, & Munawer, 1999; L. Zhang, Ahn, & Chu, 2001; X. Zhang et al., 2003).

Investigation of user to user delegation with the focus on RBAC was done in (Barka, 2002; Barka & Sandhu, 2004, 2007). In RBAC, permissions and responsibilities are associated with roles and users are made members of these roles to grant them permissions. In RBAC delegation, the user in one role can delegate his/her role to another user, so the latter can perform actions on the former's behalf. The work differentiated between permanent and temporary delegations and considered a number of cases to develop a framework for role based delegation model (Barka & Sandhu, 2000b). In another work by the same authors, they proposed the role based delegation model (RBDM0) for flat RBAC roles, and extend the work as RBDM1 for hierarchical RBAC roles (Barka & Sandhu, 2000a). The RBDM1 differentiates between upward, downward and cross sectional delegations in hierarchical roles. The upward delegation is from a junior role to a senior role and is considered useless because of the permission inheritance proposed in (Sandhu et al., 1996), where the senior role already gets all the permissions associated with the junior roles. The downward delegation is from a senior role to a junior role, which can only work with the subset of rights associated with the role because it may shrink the hierarchy of the organization. The cross sectional delegation is the most useful in RBAC as it gives the mechanism to assign rights within two different departments at two different hierarchical levels. The models also support revocation using time-out, where every delegation has a time-stamp which expires to terminate the delegation. Also, the models support grant-independent revocation, where any member of the delegated role can revoke the delegation from the delegatee.

To handle the administration in RBAC and to manage users and roles assignments in an organization, an administrative RBAC model (ARBAC) was proposed in (Sandhu et al., 1999; Sandhu & Munawer, 1999). The basic motivation behind ARBAC is to use RBAC to manage RBAC itself. The model is divided into three major components: user-role assignment (URA), permission-role assignment (PRA), and role-role assignment (RRA). URA model is used for the management of user to role assignment, and is defined in two sub models – URA grant model and URA revoke model. The first sub-model deals with the grant of role membership to users, and the second deals with the revocation of users' membership from the role. The PRA model is used for the management of permission to role assignment, and is similar to

URA. The permission role assignment and revocation are handled by 'can_assign' and 'can_revoke' relations respectively. The RRA model is used for the management of role to role assignment and distinguishes among three types of roles, namely abilities – roles consists of permissions only, groups – roles consists of users only, and UP-roles – roles consists of both users and permission (Sandhu & Munawer, 1998b). The RRA model is composed of several sub-models for ability-role assignments and group-role assignments.

The flat and hierarchical delegation models (Barka 2002; Barka & Sandhu 2000a, 2000b, 2004, 2007), and URA, PRA and RRA administration models (Sandhu et al., 1999; Sandhu & Munawer, 1999) are nice additions to the RBAC original model. However, only security administrator can control these assignments and their continuous involvement is necessary, which can increase management efforts in large distributed systems. This problem was addressed in (L. Zhang, Ahn, & Chu, 2001, 2003), where a delegation approach is investigated to decrease the load of role management from security officers. The delegation model is based on RBAC (Ferraiolo & Kuhn, 1992; Ferraiolo et al., 2001; Sandhu et al., 1996) and decentralizes the administrative process in distributed environments. The model allows the users to delegate their roles by a rule based delegation framework that supports role hierarchies and multistep delegations by introducing delegation relations. Also, a rule based declarative language to specify different policies was defined. The work shares the same purpose as URA discussed in (Sandhu et al., 1999; Sandhu & Munawer, 1999).

The notions of active and passive delegation in the context of RBAC were introduced in (Na & Cheon, 2000). Active delegation covers the cases when the delegator and the requestor are same subject (after satisfying some prerequisite condition), while passive delegation covers the cases when the requestor and the delegators are different subjects, and the requestor requests the delegation server to grant the rights to act on the delegator's behalf. The model only works with upwards delegation, where the junior role can take the rights associated with the senior role. The model consists of the delegation server and the delegation protocol, where the former decides the possibility of a particular delegation by maintaining delegation policies, and the latter deals with the methods by which a delegation can be performed.

In another work, a flexible permission based delegation model (PBDM) based on RBAC was introduced (X. Zhang et al., 2003). The model supports chained, partial and temporary

delegations, and distinguishes between role level and permission level delegations. The model particularly addresses the problem of delegating high level permissions to low level users and extends the RBAC model with three types of roles, that is, regular roles, delegation roles and delegatable roles. Regular roles are kept same as in RBAC, delegation roles are specifically used for delegating permissions, and delegatable roles are based on regular roles but their permission can be delegated to other users by creating delegation roles. The model has one-to-one mapping between regular and delegatable roles, which is handled by the security administrator, the problem of illegal flow of permissions is minimized.

An attribute based delegation model (ABDM) based on RBAC role structure was proposed to handle user to user delegation (Ye, Wu, & Fu, 2005, 2006). The authors combined the high flexibility of RBDM (Barka & Sandhu, 2000a) and medium security of PBDM (X. Zhang et al., 2003) to construct a model that works on delegation constraints. These delegation constrains consist of delegation prerequisite condition role (CR) – membership of specific roles, and delegation attribute expression (DAE) – set of attribute constraints that must be fulfilled by the delegatee. A delegation can only take place if the delegatee role satisfies both the CR and the DAE.

Some other research works also proposed the semantics of delegation and its revocation. For example, Li and colleagues proposed a logic of delegation in large scale distributed systems (Li, Feigenbaum, & Grosof, 1999; Li & Grosof, 2000). Crampton and Khambhammettu (2008) proposed a rights transfer model based on RBAC, but for completeness they also included the delegation model. The delegation proposed in their work allows the delegator to grant a role or a permission to the delegatee in the context of flat as well as hierarchical roles of RBAC. A delegation model for trust management systems in RBAC was presented in (Tamassia, Yao, & Winsborough, 2004). The model combines the advantages of RBAC in trusted systems and delegation in distributed systems. It was designed to support chain delegations and verify the hierarchical delegation chain. Another fine grained user-to-user delegation model for RBAC allows the users to delegate part of rights associated with their role to another user (Wainer & Kumar, 2005). Another delegation work for RBAC in the presence of multiple hybrid hierarchical structures was explored in (Joshi & Bertino, 2006). A trust based delegation model for pervasive computing was discussed in (Steffen & Knorr, 2005). A delegation model of

RBAC in workflow scenarios was discussed in (Bammigatti & Rao, 2008), and a delegation model for workflow management systems was proposed in (Atluri & Warner, 2005).

There are several shortcomings with the application of RBAC delegation for online social interactions. First, the delegation is not expressive enough in RBAC and the only way to grant permission to a user is by granting the permission to a role and assigning him/her that role. This leads to management overhead for administrators as running multiple delegation scenarios at the same time using roles is difficult. Another shortcoming of the RBAC delegation models is their coarse nature, as they do not differentiate among different users within the same role. Further, any user in the delegated role can revoke the delegation in RBAC, while the delegation model based on socio-technical design only allows the delegator to revoke the rights.

2.8.2 Rights transfer

Rights transfer is a process which allows a user *Alice* to give away rights to another user *Bob*, where the transferred rights are no longer available to *Alice*. It is used when there is a need to restrict the cardinality of users having the same right. In some of the literature, rights transfer is referred to as transfer delegation (Crampton & Khambhammettu, 2008, p. 1) or permanent delegation (Barka, 2002, p. 72). The work on rights transfer for traditional access control models is presented in this section.

In his PhD dissertation, Barka (2002, p. 38) proposed the user to user delegation model based on RBAC. The delegation was categorized as temporary and permanent, which can be considered as delegation and transfer in the context of the current work. In the traditional user-to-user rights delegation section[17], Barka's work on temporary delegation in the context of RBAC was discussed. Now his work on rights transfer (permanent delegation) will be discussed in this section. The transfer model was based on RBAC and allows a user to permanently transfer his/her role membership to another user in the organization, where the previous member losses his/her membership of that role. The permissions associated with the transferred role are also given to the new member. The transfer model is irrevocable, where the old member cannot take back the role membership but only the security administrator can

[17] Section 2.8.1.3.

remove the new member from the role. The work reported two permanent role-based delegation models – PRBDM0 and PRBDM1, for flat and hierarchical roles respectively.

Crampton and Khambhammettu (2006, 2008) also proposed a rights transfer model for flat and hierarchical roles in RBAC. The transfer model allows the user to grant a role/permission to another user, where the latter gets all the transferred permissions while the former can no longer exercise the permission or role. Two different relations were introduced for role and permission transfer each. The first deals with the set of roles/permissions that can be transferred from the set of all the available roles/permissions in the system, and the second deals with authorization of users to transfer a role/permission. The transfer considered was temporary in nature and can be revoked. The work is similar to Barka (2002), but considered temporary permission transfers as well. The authors reported that the only difference between delegation and transfer is monotonicity of rights, which means that *Alice* cannot exercise the right after its transfer, while she can exercise it after delegation[18].

2.8.3 Rights sharing

Rights sharing is a process which allows the user to give the right to another user while keeping it at the same time, so both can exercise it. Rights sharing is mostly used in scenarios where multiple users share the same right over an object. There are some research projects that indirectly relates to this area. One of those research works is of Y. Liu and colleagues (2009), which deals with rights sharing in online digital rights management (DRM) systems for digital contents. DRM provides the management, tracking and protection of copyrighted digital assets (Q. Liu, Safavi-Naini, & Sheppard, 2003; Iannella, 2001). There are some commercial DRM systems available from Microsoft[19], IBM[20] and Apple[21]. The online rights sharing model by Y. Liu and colleagues (2009) works on light-weighted ticket transfer protocol which does not change the semantics of DRM system. The model supports time constraints and rights description to support fine grained rights management. A security mechanism was also

[18] However the current research differentiates between delegation and transfer, on the basis of the ability to further allocate the right.

[19] Microsoft Windows Media Digital Rights Management, http://www.microsoft.com/windows/windowsmedia/forpros/drm/default.mspx, 2007.

[20] IBM's Electronic Music Management System (EMMS), http://www.almaden.ibm.com/cs/madison.html, 2005.

[21] iTune. http://www.apple.com/itunes/, 2006.

proposed to encrypt the tickets and reduce the risks of over-issuance and multiple spending of tickets.

Another rights sharing model based on digital rights management for copyrighted digital contents was explored in (Z. Zhang, Pei, Ma, Yang, & Fan, 2008). The proposed sharing of rights over copyrighted contents was mutually inclusive as all the previous and new holder of the right may exercise it. The authors proposed a granular, time, depth and cardinality constrained sharing model based on trusted computing components. However, the model is based on static trust certificates, which may not support the dynamic nature of social interactions.

2.8.4 Rights merge

Rights merge is a process which allows the user to merge the rights of a set of users, where all must agree to exercise the right. It is used in collaborative environments where multiple users' consent needs to be considered for any decision. Little work on rights merge has been done in literature for access control models. Among them, Ilic and colleagues (2007) explored a dual ownership model for access control in safety supply chains. Another approach for ownership protection along with a model for joint ownership of watermarked digital contents was explored in (Wu, 2003). In addition, Guo and Georganas (2002) proposed a joint-ownership verification algorithm to verify the watermark on digital images. However, all of these approaches were more oriented towards verification of ownership rather than rights allocation over the online resources.

In traditional access control models, only delegation has got some serious attention but other modes of rights allocation, for example, rights transfer, rights sharing or rights merge are sparsely explored. Moreover, the work of rights allocation in literature is less useful if applied to STS due to the following reasons:

The number of users and objects is quite high in STS as compared to traditional systems. For example, Facebook reports 955 million active user accounts, each adding many hundreds of photos and comments each year, with more than 100 billion friend connections; so models that map system resources to all the users in the system soon become over-complex for a STS with millions of users and billions of resources. Role based access control (RBAC) is often

proposed as the solution to reduce the complexity of traditional systems by introducing roles (Ferraiolo & Kuhn, 1992; Ferraiolo et al., 2001; Sandhu et al., 1996). However, access control for STS depends on the number of interactions rather than the number of users in the system, so its complexity increases geometrically with size, not linearly. Also, the object visibility in STS is restricted to the social circle of the owner rather than the whole system, which makes traditional systems unsuitable for them. Furthermore, the frequent content updates and the volatile nature of friendship makes it difficult to use traditional access control systems for STS (Hart et al., 2007).

Connecting to others satisfies relationship needs but also raises security and privacy concerns (Simpson, 2008). As users contribute online contents, for example, family photos or online videos, they naturally expect to control those contents. In centralized access control, each user is allocated the same access control policy values, so variants must be requested from a central authority (Hart et al., 2007). Roles in traditional access control systems are system wide groups whose membership is set by a system administrator. The access rights over user's resource are allocated to a role the user has no control over. In contrast, users involved in online social interactions want to set their own values without reference to a central authority, for example, to share their data with everyone or to restrict it to family and friends. Traditional access control systems do not provide all the functionality needed and so struggle with the demand of today's individual user for a diverse range of privacy requirements (Gollu et al., 2007, J. Li et al., 2009). Generic roles tend to reveal more than the users want, as they cannot specify local requirements using generic roles.

In addition, the set of operations offered by STS is much larger than the number of operations offered by the traditional systems. To a traditional access control system, a file can only be read, written to or executed, but users involved in online social interactions want to exercise a much richer set of operations in parallel to that used in physical societies. This sophisticated set not only includes read, write, vote, tag, subscribe and so on, but also other operations to manipulate different types of objects with different collaborations among different users, such as joint ownership of a couple common online persona, transferring a colleague's research paper to him/her, or delegation of management of tracks to track chairs from conference organizers. Rights allocation is useful in many situations, such as backup of role, collaboration of work and decentralization of authority (X. Zhang et al., 2003; Park & Lee, 2005).

STS are built around the social requirement of ownership, which is not widely supported by traditional access control models. From the set of traditional access control models, only DAC can work with the ownership requirement of STS, but it faces the problems of global visibility of objects, central administration of groups, and users' interactional complexity. Another alternate to the complexity problem is RBAC, but it is proved to be even more expensive while supporting ownership (Sandhu & Munawer, 1998a). Besides, traditional access control models assume single ownership of objects, whether it is a person or an organization, but do not support multiple-ownership of objects which may restrict online social collaboration opportunities.

Further, only one of type of rights allocation – rights delegation is extensively explored in literature but other rights allocations, such as rights transfer, rights sharing and rights merge are hardly explored in much detail. Also, the rights delegation work in literature is mainly about role to role delegation in RBAC, where permissions are associated with roles. However, STS require domain based rights delegation where permissions are associated with objects. Besides, the delegation models in traditional access control do not support multiple sub-delegations, but provide support for multi-level delegations – delegatee can further delegate its role, which raises the question of accountability in online systems. On the other hand, STS require single-level, multiple delegations to maintain the accountability and to support the coexistence of multiple delegation subdomains, for example, one conference can be divided into multiple tracks, each of which can be then delegated to different track chairs. Another major difference between traditional delegation models and the required one is the scope of allocation. In traditional models, the allocation scope is system wide, while the required allocation may only have domain based scope and will be invalid outside the domain and outside the local role's scope[22]. Figure 2.2 illustrates the progress of access control and identifies the position and contribution of the proposed work.

This research revisits the problems of access control in STS using socio-technical design and introduces an access control model supporting rights allocation as well as ownership and relationships. There is currently no formal access control model for STS that supports rights delegation, rights transfer, rights merge or multiple-ownership.

[22] Which restricts the demage of allocation if anything goes wrong.

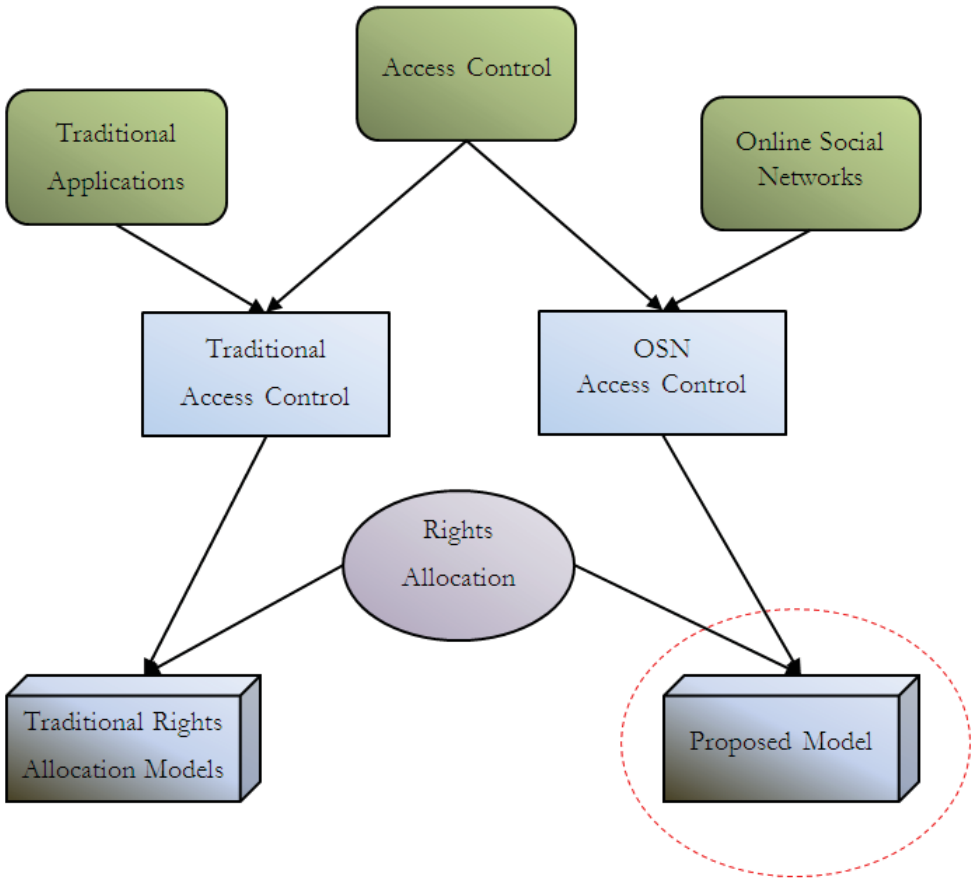

Figure 2.2: Summary of the literature and contribution of the current research

2.9 Thesis statement

From the literature review, the following conclusions can be derived:

a) STS are new and their emergence has introduced one of the largest user interaction platforms that computing has ever seen. They are unlike any of the previous existing platforms, and have technical as well as social requirements.

b) Various access control models for STS have been proposed in recent years but they lack a systematical scheme for managing rights allocation, including rights delegation, rights transfer, rights share and rights merge.

c) While traditional access control models can be used for this new type of software, and some types of rights allocations are already explored for them, their support to capture the STS requirements is difficult for several reasons:

    1) Traditional access control models define the access rights for every resource for every user/role in the whole system. However, this approach is too

complex for STS as it contains millions of users and billions of resources. Also access control for STS depends on the number of interactions, which makes access control entries quite enormous.

2) Traditional access control models work on global administration and roles are system wide groups whose membership is set by a system administrator. In contrast, users involved in online social interactions require local control and domain based local roles without any central reference.
3) Most of the traditional access control models do not support ownership. Also, in cases where they do support it, they do not have fine grained policy implementation required for STS like fine grained local roles and their membership. Further, the opportunities and challenges associated with multiple-ownership are raised in STS are not much explored before.
4) The rights allocation models proposed in literature were mainly about role to role allocations, as permissions were associated with roles. However, STS require domain based rights allocations where permissions are associated with objects.

Hence, the existing work in rights allocation is not suitable for STS for their differences in nature with the traditional applications.

2.10 Research question

Given the above thesis statement, this research hypothesizes that a general access control model can be designed to supports rights allocation in all online social interactions. The model must also support social requirements like ownership, local relationships and heterogeneous privacy policies. The objective of this research is to answer the following question:

Q) Can a general rights allocation model for online social interactions, which is decentralized, logically consistent, socially valid, and supports dynamic local roles, be designed?

This question also raises the following issues:

a) What are the basic types of rights allocation for online social interactions?

b) Can the allocation models be applied to simple rights as well as meta-rights?
c) What are the characteristics of those rights allocations?
d) What are the rights of various roles associated with an object after every allocation?
e) What is the difference between the allocation models for simple and meta-rights?

Also, the rights allocation model has the following base condition:

Condition) Can an access control model for STS be designed to support ownership, relationships and local administration?

a) Can the model be applied to emerging scenarios of online social interactions?

2.11 Methodology

A methodology is defined as a set of procedures to meet some predefined goal (Welke, 1981). Sometimes it specifically refers the actual research method used for some research (Tashakkori & Teddlie, 1998), and recommends the sequence in which the procedures should be performed throughout the lifecycle of the project (Heuvel, 2002). Methodologies are also considered as world views as they make certain assumptions about the nature of the paradigm (Kuhn, 1970; Burrell & Morgan, 1979). Therefore, different methodologies generate knowledge about different aspects of the same world (Mingers, 2001). This research aims to design a rights allocation model for online social interactions; therefore it follows the constructive research methodology which focuses more on developing novel technologies and innovations (Crnkovic, 2010).

2.11.1 Constructive research methodology

The constructive methodology is a research approach for producing innovative constructions, intended to solve problems faced by real world and thus makes contributions to the theory of the applied discipline (Kasanen, Lukka, & Siitonen, 1993; Lukka, 2003; Caplinskas & Vasilecas, 2004). The focus of constructive methodology is more on developing new models, algorithms and design principles rather than solving a local problem in a new way (Glass, Ramesh, & Vessey, 2004). This research aims to produce a rights allocation model for online social interactions. The model is divided into four sub-models, namely, Replace, Share, Merge and Revoke, to handle different types of rights allocations. These models may solve the problem of

rights allocation in online social interactions and may also contribute to the theories of access control, ownership and online social interactions.

This research adopts the constructive research methodology, where the construction of the models starts with identifying the characteristics associated with each of the models. These characteristics are then refined based on the parametric values, and the procedural steps are defined for the implementation of each of the models. Further, the effects of each model application on different roles associate with the allocated object are analyzed along with their design principles. The construction provides the logic for the development of rights allocation models for online social interactions that can be implemented in any programming language over any platform.

The constructive research methodology used in this research divides the research process into the following phases (Kasanen et al., 1993):

a) Problem Identification: Absence of rights allocation model for STS.
b) Literature Review: Detailed understanding of traditional access control models, types and requirements of STS, their access control models and current practices, along with various rights allocation models for traditional systems.
c) Conceptual Framework: Identification of STS access control components and their types, along with the basic rights allocation operations and their characteristics.
d) Innovation: Detailed specification and modeling of the rights allocation models.
e) Demonstration[23]: Show that the proposed solution works on both actual and hypothetical cases of different types of current STS.
f) Contribution: Reporting theoretical connections and research contributions.
g) Applicability: Examine the scope of applicability of the solution.

The basic illustration of the general methodology used in this research is shown in figure 2.3.

[23] In modeling methodologies, the model's working is usually demonstrated. As this research designs the model for online social interactions, it could not be demonstrated without building the community first. So, the researcher has demonstrated the model on actual and hypothetical cases from current online social interaction scenarios.

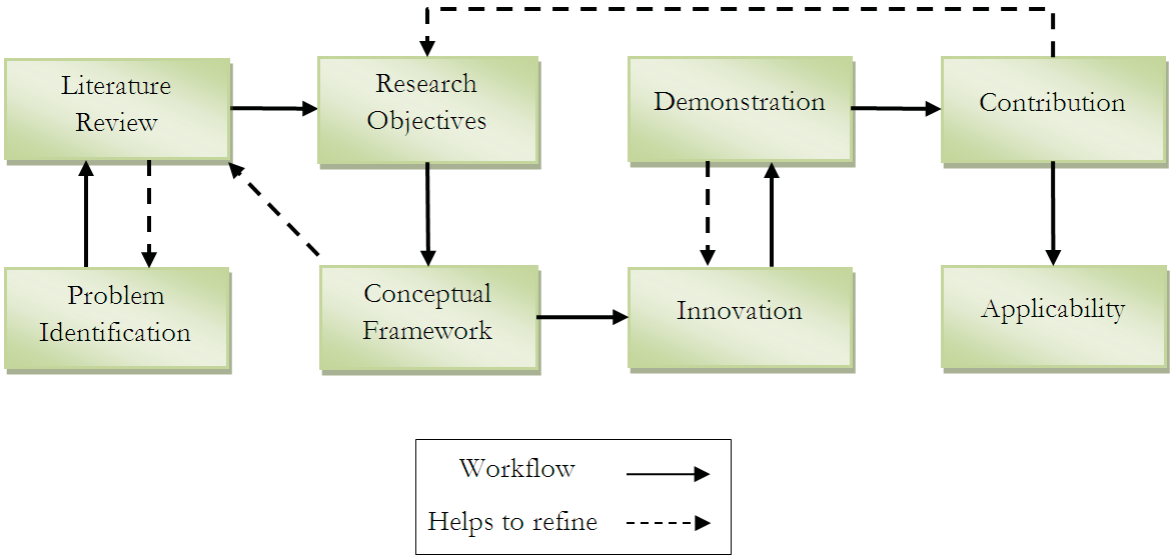

Figure 2.3: The general methodology used in this research (adopted from Kasanen et al., 1993)

This dissertation follows the constructive methodology in the following manner: The background and motivation for the problem are explained in chapter 1. Chapter 2 outlines the literature related to the research and concludes with research question and objective. Chapter 3 describes the conceptual framework along with the reduction tree for the identification of basic rights allocation models. Chapter 5 and 6 illustrate the innovation phase by outlining the use-rights and meta-rights models in detail. Chapter 8 demonstrates the working of the rights allocation models by showing that they work on current as well as on hypothetical use-cases, while chapter 9 emphasizes on the contribution and applicability of the presented research.

The core element of a successful constructive methodology is the innovation phase, which is often heuristic by nature. However, this research has used an incremental approach (Mills, Dyer, & Linger, 1987) to devise each allocation model, and extreme formal modeling (XFM) within it to refine the rules and characteristics by applying them onto various online social interaction scenarios.

The use of XFM has several benefits over the ad hoc abstract modeling. First, the ad hoc construction is error prone and the effort of model building and debugging grows exponential along with the size of the model, whereas XFM includes one property at a time and an error in the model automatically identifies the latest property as the culprit. Second, the ad hoc approach has the tendency to include more behavior than the specifications which increases unwanted complexity, but XFM, by definition, is less likely to include extra features. Third, in

ad hoc model construction, there is no way to confirm the inclusion of all properties, thus significance of the model reduces. However, XFM incrementally adds properties, so it constructs the model having all the functionalities.

2.11.2 Extreme formal modeling (XFM)

Extreme Formal modeling (XFM) (Suhaib, Mathaikutty, Shukla, & Berner, 2005) is an agile formal methodology based on extreme programming concepts to construct abstract models from natural language specifications of the complex systems. It was designed to bridge the gap between formal system engineering and requirement specification. It provides a 'golden reference model' that is not only correct with respect to the requirements but also unambiguous through formal semantics. XFM ensures that the model is regressively verified during the construction phase, so the resultant abstract model remains constructively correct and closer to the intended specifications. This is achieved by adding one feature at a time and testing the whole model after every increment.

The initial phase of XFM involves the breakdown of intended outputs of the model. Then starting from the basic functionality of the system, it takes one functionality at a time and transforms it into some logical property. It is important to focus on the behavior of the current property while constructing the model. The next step is to check if the logical property correctly expresses the intended output. Once the property satisfies the specific functionality, the next functionality is taken, converted into a logical property, and is tested against the intended output to check whether both properties hold. On the other hand, if the property does not satisfy the functionality, it is debugged until it matches the intended output. This procedure is repeated until the abstract model acquires all the functionalities of the intended behavior. The result of this incremental approach is a compact and structured abstract model. Whenever the model fails, the most recent addition is investigated as all previous properties were well structured and tested.

This research first divides the rights allocation into four basic sub-models, that is, Replace, Share, Merge and Revoke, and applies the XFM to one model at a time. For this purpose, it takes a model, explores its different functionalities, convert them one by one into logical properties and incorporate into the model. At each step, it ensures that the added functionality exhibits the desired function. After completing all the functionalities of one model, it then

moves onto the next one. The complete set of models are then tested and demonstrated on both real and hypothetical online social interaction use-cases. The details of the use of XFM for this research are illustrated in figure 2.4.

2.12 Summary

This chapter has summarized the access control literature, its evolution in design and outlines various well-known access control models for traditional applications. It has acknowledged the fact that access control is an important component of every computer system and is designed against the particular requirements of the application. It has further highlighted the emergence of various well-known requirement sets and the models based on them. This strengthens the fact that the design and requirements of computer applications are different from each other and one access control model cannot fit the requirements of all types of applications.

Following this basic description of access control models, the chapter further highlighted the emergence of STS, their properties and the issues related with them. STS are based on social requirements of physical communities and so share their inherited structure. The users' interaction, their social circle, the desire to own and trust are some of the key features of these STS and the access control models should incorporate these social requirements in their design as well. After outlining the requirements of online social interactions, it has briefly described some access control models for social networks.

As there does not exist any rights allocation model for online social interactions, the chapter then presented some previous work on rights allocation in traditional access control models on rights delegation, rights transfer, rights share and rights merge. The delegation work is explored in much detail in literature and categorized into three types: machine to machine delegation, user to machine delegation, and user to user delegation. However, other types of rights allocations, that is, rights transfer, rights share and rights merge are not explored in much details.

After presenting all the access control models for social networks and rights allocation models for traditional applications, the chapter outlines the differences between their structures and the reasons why they cannot be mapped onto the new requirements of online social interactions. The reasons include number of users', their interactional structure, local

administration, local object visibility, support for ownership and the nature of allocations. This problem and the differences among traditional access control models and STS lead towards the research question. Further, the constructive research methodology was presented as an approach to answer the research question. The next chapter will cover the basic components of an access control model based on socio-technical design.

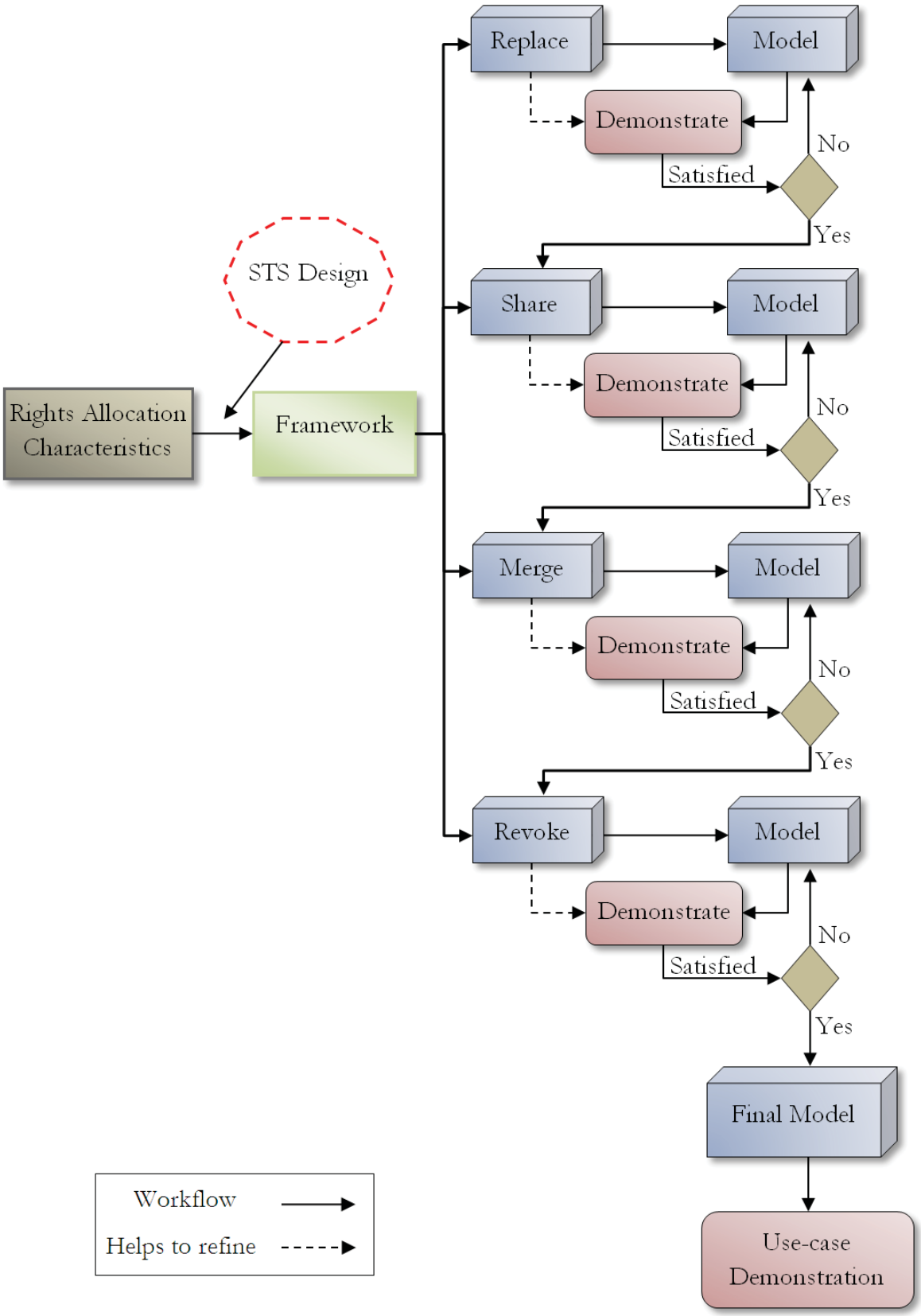

Figure 2.4: Detailed illustration of the innovation phase using incremental Extreme Formal Modeling (XFM)

Chapter 3

Conceptual Framework

To answer the questions raised in chapter 2, this chapter outlines the basic components and the possible ways to use them. The conceptual framework is based on the socio-technical design, which involves technical and social requirements, to design not just what *can* be done but what *should* be done. This chapter is organized as follows: first it outlines some basic characteristics of online social interactions by generalizing the literature explored in chapter 2. It then describes some basic concepts used in access control literature for designing various models. This basic discussion leads to rights allocation, which introduces various methods to allocate rights. It further presents various characteristics of rights allocation followed by the reduction approach that provides the basis for rights allocation models.

3.1 Characteristics of online social interactions

Generalizing the work on access control models for social networks from the previous chapter, this section picks their common characteristics. However, before going into these

details, it illustrates an example of a social network[24] (for example Facebook), which can help in highlighting the properties and will direct the design of an access control model for STS.

Figure 3.1 shows a simple social network where nodes are network users and lines are the relations between them. These relationships can be of different types, for example *Alice* sees *George* and *Frank* as friends, *David* as family, and *Bob* and *Carl* as colleagues. These direct friends of *Alice* further see other users as friends, which are considered as indirect friends of *Alice*.

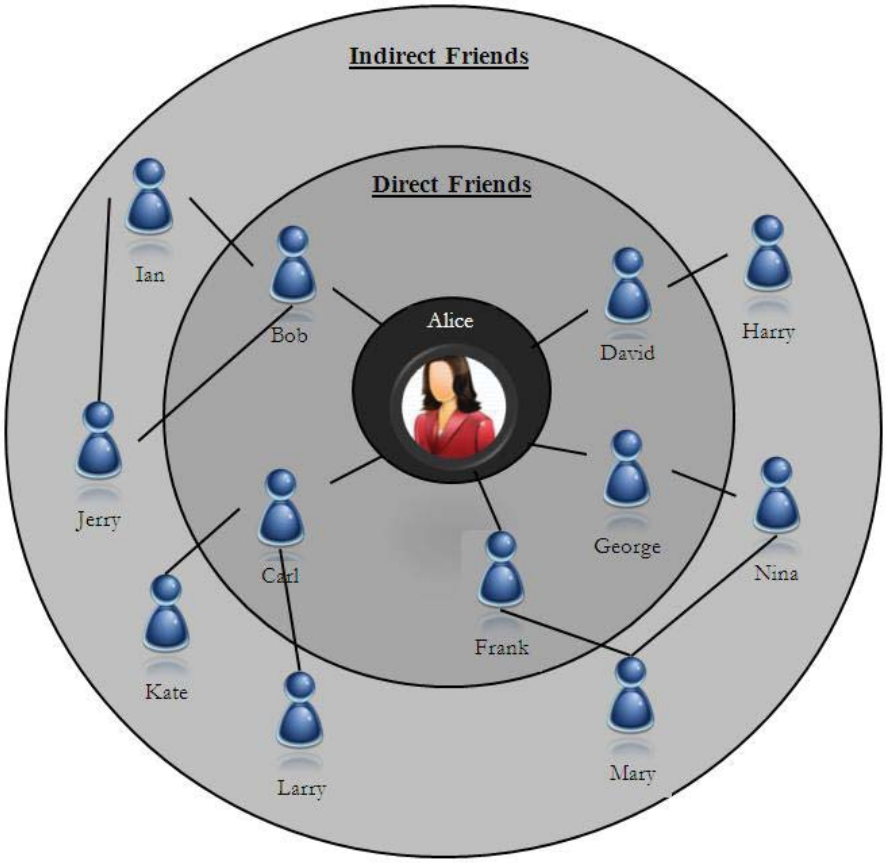

Figure 3.1: A simple social network

In social terms, if *Alice* owns some object, say a photo, only she has the right to manage its access (Locke, 1963). If she then shares it based on her personal connections, it is not necessary to map that resource to all the users beyond her social circle. This introduces the

[24] Social Networks are purely based on social interactions and support its all three types, so can be seen as the richest example of STS having all types of online social interactions.

concept of local visibility, which is used by modern access control models for social networks to reduce the complexity (Carminati et al., 2009; Fong et al., 2009). In the social circle, relationships have degrees of closeness as people relate not only to their best buddies but also to family, coworkers, teachers and acquaintances, and access to resources is managed based on the type of relationship with other users (J. Li et al., 2009; Tapiador et al., 2011).

Equally, sharing of the photo can take many forms as it can be *Alice* photo which she wants to show her friends, it can be her photo with *David* on which she wants to tag him, or it can be some campaign poster or community service message which she wants her friends to propagate. These types of activities are supported by rights allocation, which is useful in many situations such as backup of role, collaboration of work and decentralization of authority (Park & Lee, 2005). In physical communities, lending a book to a friend, selling/purchasing of car or house, joint ownership of bank account by a married couple are some common examples of rights allocation. To evolve socially, a system must allow rights to be allocated, given an initial state of just one system administrator holding all rights and can be done by involving others in ones permissions (Gaaloul et al., 2008).

The above mentioned requirements of social networks can be outlined in details as follows:

3.1.1 Creator Ownership

In the physical society, people see objects in terms of who owns them (Locke, 1963). In particular it is considered right that one should own what one creates, for example a painting or poem. In STS, people create information objects by posting them, so they should own them, that is, be able to manage their access. Essentially the privacy requirements of STS are that if people cannot control access to what they post, they will not post, and so the system will collapse (McInnerney & Roberts, 2004). The concept of ownership divides the whole system into multiple autonomous administrative domains, managed by owners themselves. STS are built upon the social concept of ownership, that is, every resource in them must be owned by at least one person. Ownership can also add complexity, as in physical society it can be sold, delegated and shared.

3.1.2 Freedom

In the physical world, freedom is the right to control one's body, to not be a slave to another. If freedom online is the right to control one's online body, or persona, one should be able to edit or delete it, yet many systems don't permit this (Lessig, 1999). A system offers freedom if actors can remove themself from it, for example delete a Facebook wall or YouTube channel with nothing left behind. The social logic of freedom is that one owns oneself online, that is, an online persona does not belong to the system administrator (SA).

3.1.3 Privacy

Privacy can be defined as the right to control the access of others over ones information (Gavison, 1980; Allen, 1988; Moore, 2003). The boundaries of privacy differ among individuals and cultures, but share some common grounds. Due to this diversity, a system offers privacy if actors can devise their privacy policy and the system helps them to impose it, for example a Facebook photo can be shared with a couple of friends, with one's whole social circle or with the entire community.

3.1.4 Relationships

Every user in STS has a unique social circle which may consists of her friends, family and colleagues[25]. This social circle is different for every user where they define their own privacy requirements, for example a person may trust their friends more than their family and so give them more privileges. So it cannot be system wide; rather its scope is just within the domain of the owner. Similarly, as people are not only connected to their friends but also with their family, colleagues, acquaintances and others, the social circle should not be coarse but fine grained which can be divided according to the closeness in relationship between the two users (J. Li et al., 2009; Tapiador et al., 2011). Furthermore, as the relationship between two users changes over time[26] and asymmetric in nature[27], the support of relationships in social networks should also be dynamic and asymmetric. This implies that relationships in social networks should support unique, fine grained, dynamic and asymmetric social circles. These

[25] It can be extended to involve general public.

[26] One may add some user as acquaintance which evolves into friend and close friend over time.

[27] Alice may consider Bob just as colleague but he may consider her as friend.

requirements can be met using domain based local roles where the owner configures them according to their privacy requirements. Roles and access rights should vary by local domain, that is, let users define local roles for their domain to decentralize resource management. As the scope of these roles is limited to the domain, they can provide support for dynamic asymmetric relationships.

In contrast, current access control models for social networks do not distinguish between friends, and treat all of them with the same privacy policy, where users can share their information either with level one or with level two members (Fong et al., 2009). Also, the supported relationship is symmetric (Ali et al., 2007), and static in nature (J. Li et al., 2009; Tapiador et al., 2011).

3.1.5 Objects' local visibility

In social networks, traditional access control models face a serious scalability problem, as potentially many more subjects must be mapped to many more objects, regardless of whether a subject has access rights over an object or not. A special matrix is used to estimate the relations between subjects, objects and operations by $S \times O \times opr$, where S is the number of subjects, O is the number of objects and opr is the number of operations possible on that object. The authorization matrix

$$subject \times object \times operation \dots [eq.3.1]$$[28]

is therefore huge and diverse (Kerschbaum, 2010). For example, currently Facebook reports to have more than 955 million active users with 90 resources added by each user, every month[28]. In a traditional Discretionary Access Control (DAC) model, this is over 246 trillion access control entries per month, where every request must traverse the whole list. One often proposed solution is Role-Based Access Control (RBAC) (Sandhu et al., 1996), which succeeds in reducing complexity (Chung, Choi, Lee, & Rhyoo, 2007; H. Lee, K. Lee, & Chung, 2006; Wu, Ke, & Tzeng, 2008) by dividing the authorization matrix using a level of indirection via the role concept:

$$subject \times roles \qquad role \times object \times access \dots [eq.3.2]$$

[28] Facebook statistics page, http://www.facebook.com/press/info.php?statistics, accessed on 30th July 2012.

The subjects are mapped to roles which are assigned access permissions over objects. In this way, the access permissions for thousands of subjects are reduced to hundreds of roles. However, RBAC cannot be used for STS due to system wide roles, and their global management. Also as discussed earlier, online social interactions need the ownership support while RBAC has proved to be even more expansive than DAC if ownership is supported (Sandhu & Munawer, 1998a).

Current access control models for STS introduced the local visibility of objects by introducing social circles. Users mostly share their information among their social circle, so it is not necessary to map ones resources with all the users in the system. This reduces the authorization matrix, as the visibility of an object is not across the whole system but limited to the social circle of each owner. This limits the number of subjects having potential access over resources:

$$Potential\ Users\ = Subject\ \cap Social\ Circle \ldots [eq.3.3]$$

where subject is the number of users present in the system, social circle is the number of users connected to the owner, and potential users are the users that can request a resource from the owner.

The concept of restricting the visibility of objects to the social circle of owner is quite useful in reducing the complexity of traditional access control models. As objects are not mapped to the whole user set of the system rather to only a small subset of it, the visibility of resource reduces to local roles. Thus, the authorization matrix for the whole system is reduced to

$$\sum_{i=1}^{N}\left[\sum_{j=1}^{x} Friends_j \times \sum_{k=1}^{y} Objects_k \times \sum_{l=1}^{z} Operations_l\right]_i \ldots [eq.3.4]$$

where N is the number of total users present in the system, while x, y and z are the number of friends, objects and operations that exist in one's social circle. This local visibility reduces the access control entries from 246 trillion to 33 trillion for the above mentioned data, and is illustrated in figure 3.2.

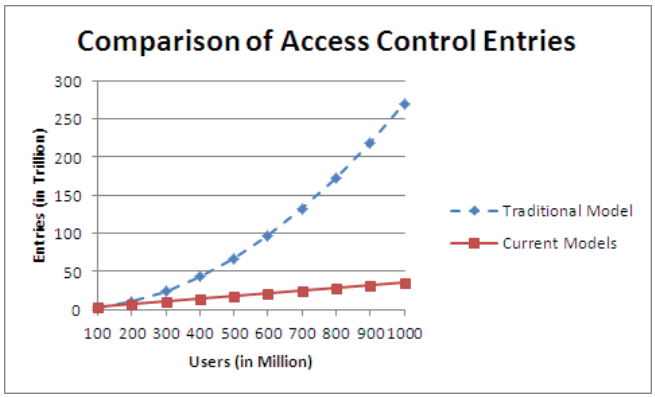

Figure 3.2: Access control matrix magnitude for different models based on eq. [3.1] and eq. [3.4] for Facebook statistics

3.1.6 Object classification

Most of the current access control models treat objects as independent entities, which are managed separately regardless to their disclosure level. This approach is quite simple but lacks the ease of use and access efficiency which may result in poorly configured privacy policies. On the contrary, most social networks have different types of resources where some are more private than others, like a family photo as compared to activity status. So resources of the same disclosure level can be grouped into same clearance class[29], which can be given a privacy classification, for example to let only family view the family photo album. The classification of objects has only explored in some access control models such as (Hart et al., 2007), where access control decisions are based on the contents present in an object. Creating owner oriented object classes to define privacy levels reduces rights management complexity and increases usability.

3.2 Access control specifications

As most of the online social interaction requirements have been outlined based on socio-technical design, this section now gives the basic understanding about access control components. The standard view point of an access control system is a composition of set of subjects, set of objects and a set of operations (Lampson, 1972).

[29] For example, a photo album.

3.2.1 Actors

Any human interaction system must contain users (or subjects) that represent people or a group of people who exist in that system (Whitworth, de Moor, & Liu, 2006). They may be referred to people interfacing with a single computer system (Ferraiolo et al., 2003), or communicating with multiple communication forums (Kling, McKim, & King, 2003). The relationship type can also be multiple as it may be an instance of user's particular login (session), it may be a computer agent working on user's behalf, or a user associated with a role (RBAC) (Sandhu et al., 1996).

This research uses the term 'actors' rather than 'users' or 'subjects' for several reasons. First, a user is characterized in relationship to a particular system, whereas an actor may participate in multiple communications forums (Kling et al., 2003), such as an e-journal, an e-publisher, and several conferences. Second, the term 'user' generally implies a single type of relationship with the system, whereas an actor may play multiple relationships (Kling et al., 2003), and he may have conflicting or ambiguous requirements about the actions they perform (Lamb & Kling, 2003). For example, a reviewer in an e-journal system could also be an author. Also, subject is a bit more system term, where it means any computer process[30] requesting a resource (Gasser & McDermott, 1990), while user is more HCI term which refers to users' choices, their cognitive process and other HCI factors (Myers, 1998). However, actor is a more STS term where an actor exists within a community and can have many types of relationships in a system.

In general, the actor is any entity that can participate in a social interaction on a user's behalf[31] and can be of two types:

a) Persona: An online persona represents an offline party, for example an avatar, profile, mail account, Facebook wall or YouTube channel can represent an offline person, group or organization. An online persona is activated by a logon operation, which equates it to the offline party and makes that party accountable for the persona's actions. An online computer agent can act for a group, like installation software for a

[30] May be on user's behalf.

[31] An actor needs not be a person, for example a program can be an agent.

company, but social acts must ultimately trace back to people and online is no different[32]. If an installation misleads, we sue company directors not software[33].

b) Group. A set of personae acting as one.

3.2.2 Objects

Objects are passive entities that convey meaning, that is, evoke cognitive processing, for example a photo. They are manipulated by operations (Lampson, 1969), and can be categorized into items and spaces.

3.2.2.1 Items

An item is a simple object with no dependents, like a board post. It can be viewed, edited or deleted. If the system object hierarchy was a tree, its leaves would be items. Items can be of different types, for example:

a) *Comment*: An item whose meaning depends on another, for example *'I agree'* makes no sense without a source item.
b) *Message*: An item with sender/receiver, for example an email.
c) *Vote/Like:* An item that conveys a choice position to a response set.

3.2.2.2 Spaces

As leaves need branches so items need spaces to carry them, for example an online wall that accepts photos or notes. In information terms, a space is a complex object with collection of dependent entities. It can be viewed, edited or deleted as an item, but can also contain other objects, for example a bulletin board. Spaces within spaces give object hierarchies, with the system itself the first space.

A space is a *parent* to any *child* entities it contains, as they depend on it to exist. Deleting a space deletes its contents for that reason, for example deleting a board deletes its posts. Allowing spaces improves access efficiency, for example one can deny access to every object in a space by denying entry to the space. An access control system can assume that every entity has a

[32] Registering by a nickname online instead of one's real name denies accountability offline but not online, for example a banned EBay seller name loses its online reputation.

[33] A person who acts as an agent can still be held accountable, for example if told to shoot someone and does do.

parent space[34]. Its *ancestors* are the set of all spaces that contain it, up to the system itself, as the first ancestor. Equally the *offspring* of a space are any child objects it contains, their children, and any other derived children.

3.2.3 Operations

Operations are actor initiated methods that target information entities. They can be clustered for access control purposes, that is, operations of the same type can be combined to simplify the design of an access control model, for example edit and append can be grouped under one set, edit. As variants of a set present the same access control issues, so to resolve one is to resolve all and an access control model that can manage one can manage all. Following are some of the common operation sets for online objects:

a) *Create:* Create adds a new entity, for example creating a Wikipedia stub for others to edit. Duplicate[35] is a variant of create.
b) *View.* Operations like view are null acts that do not change the information stored in the target but only allow others to view it.
c) *Edit:* It is the operation to change existing entity values. It has some variants like edit alters entity values, append extends them, version edits with backup, and Wikipedia's revert is the inverse.
d) *Delete:* It is the operation to remove the object and all the rights associated with it. Its variants include delete, destroy and so on, in a way that delete flags an entity for destruction, undelete reverses that, and destroy kills it permanently.
e) *Process (execute):* It is the operation to extract the basic information related to an entity to do the statistical analysis without revealing the identity of the entity.

3.2.4 Rights

In physical society, rights are defined as entitlements to perform certain actions or be in certain states (Encyclopedia of Philosophy, 2007). Communities, by norms, laws or culture, grant citizens *rights,* or social permissions to act. Rights reduce physical conflict, as parties who agree

[34] Except of course for the system itself.

[35] Creating with same parameters values (copy and paste).

on rights do not have to fight. Rights arise when social requirements manifest as personal cognitions, which manifest as informational rules, which manifest as action directives.

Physical society specifies rights in contracts, laws and customs, but in online society a right is a stored access control permission. Online, the access control program *is* the law (Lessig, 1999), but it can still act on behalf of a community, that is, it need not be a tyrant. A social right does not require the act it allows, for example the right to sue does not force one to sue. Likewise, access control model for online social interactions is about defining legitimate choices – individuals still choose what they do. Access control defines what online actors *can* do not what they *must* do, so is today more about access than control. It gives a rights context, but how people actually act depends on bias, habit, reward and the acts of others, as in any society (Rand & Branden, 1964). A right transmitted or stored is often called a permission.

A right is a system permission for actor *(A)* to apply operation *(opr)* to object *(O)* (Whitworth & deMoor, 2002, 2003), or in symbolic terms:

$$Right(Actor, Object, Operation) \quad or \quad R(A,O,opr) \ldots [eq.\,3.5]$$

The above equation can be generalized to write:

$$Right(Actor, Entity, Operation) \quad or \quad R(A,E,opr) \ldots [eq.\,3.6]$$

where actor is any social entity, for example a persona, the entity can be an object, an actor or a right, and the operation is any that is available to that entity, for example view/edit for items and enter/exit for spaces. It means that rights are not only used to apply operations on objects, but can also be used to apply operations on actors, or rights. So an actor can act on itself using a right which can be owned by him or the system administrator[36]. Also a right can be used to manipulate a right which is often referred as meta-right in literature.

The above discussion gives the following definitions:

a) *Right:* A system permission for an actor to operate on an entity. A right can be categorized in one of the following:

[36] For example, some systems don't allow the users to delete their profile.

i. *Use rights*[37]*:* Rights to act on object or actor entities.

ii. *Meta-rights:* Rights to act on right entities.

3.2.4.1 Meta-Rights[38]

A meta-right is the right to allocate a right, and defines who can operate on rights entities within the access control system (Dewan & Shen, 1998; Indratmo & Vassileva, 2007; Mattas, Mavridis, Ilioudis, & Pagkalos, 2006). Meta-rights are absolute in nature as there are no meta-meta-rights, and they can use to operate on entity rights as well as themselves.

In other terms:

$$MR(Actor, Right, Operation_{Allocation}) \ldots [eq.\ 3.7]$$

where *MR* is meta-right and the entity acted on is a right to an entity.

This is the same form as equation [3.6] but the entity acted on is now a right, and *Allocation* is any operation on a right. While dealing with modification of rights (and meta-rights), there are two basic questions that need to be answered:

a) Who can modify a right?
b) How a right can be modified?

The next section – authorization authority, answers the first question, while the second question is addressed in section 3.2.3 (operations on rights).

3.2.5 Authorization authority

This section discusses the different possibilities of authorization authorities and chooses what is more appropriate for online social interactions. There are two basic types of authorization authorities, which are considered appropriate for most of the computing applications, that is, central administration and ownership.

---

[37] Use rights are rights to use an object/actor (for example view, edit, delete) and are sometimes referred to as simple rights – operations on objects.

[38] Simple rights are already discussed as rights over object. Now this section discusses meta-rights.

3.2.5.1 Central administration

Central administration allows the organization to manage the access rights for all the users in the whole organization through some centralized mechanism. The system has one or more system administrators[39] who control every system resource and manages access for all the users in the system over them. It also covers the cases where some predefined rules compute the access decision.

Access control model that support central administration normally falls in the broader category of Mandatory Access Control (MAC) and has many useful practical scenarios. For example, access control of this sort is normally used in organizations (Brewer & Nash, 1989), military (Clark & Wilson, 1987; TCSEC, 1985), health services (Morchon & Wehrle, 2010) and many others (Jajodia & Sandhu, 1991; Sandhu & Chen, 1998; Qian & Lunt, 1996).

3.2.5.2 Ownership administration

This type of authorization is based on Locke's idea that one should own what one creates, whether a book, a painting or a crop (Locke, 1963). It allows the creator of a resource to have all the rights over it including the right to manage those rights, that is, meta-rights.

Access control model that support ownership normally falls in the broader category of Discretionary Access Control (DAC), which was developed for personal applications and held user responsible for managing the security of their data. It also has many useful practical scenarios like operating system (Belani et al., 1998), peer to peer file sharing (Bram, 2003), grid computing (Thompson et al., 1999) and many more (Freudenthal et al., 2002; Kerschbaum, 2010).

The physical society expresses rights in terms of ownership (Freeden, 1991), so specifying who owns what online can specify rights in a way that designers can support and users can understand (Rose, 2000). People want to contribute personal stuff without worrying about its unauthorized disclosure**.** So, everything posted on a STS should be owned, and conversely if people own their posts, they should manage their access. It is also one of the important features of online social interactions as discussed in section 3.1.1. Ownership of newly created

[39] Security officers.

online objects is critical to STS success and this research will use owner as the authorization authority for the same reason.

3.2.6 Operations on rights

As a single right consists of three variables, that is, actor, entity and operation by equation 3.6

$$Right(Actor, Entity, Operation) \qquad or \qquad R(A, E, opr)$$

it gives that modification of a right includes:

a) Modifying the entity: As an entity can be an object, an actor or a right, it can be modified by:

   i. Modifying the actor at the entity location means modifying the properties associated with the online persona.
   ii. Modifying the object means modifying the object set for the right or the properties associated with an object.
   iii. Modifying the right at the entity slot means changing its actor, entity or operation slot. It again has all the three levels[40].

b) Modifying the actor set of a right over an object[41]

   i. Replacing an actor with another for a right over an object.
   ii. Adding some actor to the right and the rights of existing actors do not change.
   iii. Merging the rights of all the existing actors for an object.
   iv. Removing some existing actor from a right over an object.

c) Modifying the operation that is applicable to that entity

   i. Addition/removal of some operation associated with a particular type of entity.

This research does not deal with the modification of operations as they change according to the object type, and new operations came with new object types. For example subscribe

[40] It's logical recursion, consider modification of actor set for assigning rights.

[41] Which deals with allocating rights to actors, that is, the meta-rights.

operation is associated with mailing lists and event subscriptions, like/unlike operation is used over photos, videos and status, and announce operation is used when you want everybody in your friend list to know about some special event.

Besides, this research does not deal with the modification of object in the entity slot of the rights triplet as it is associated with modifying its properties, which are different for each object type. For example, properties of a video, picture and comment are different from each other and depend on the application type. Also, it does not deal with the modification of actor in the entity slot of the rights triplet as modifying the actor properties is associated with particular actor type, for example the properties for email profile, Facebook personae, and YouTube channel are different from each other and should be explored for the specific application type.

However, this research deals with various ways to modify the actor set of a right, that is, ways to assign rights[42]. As discussed earlier, the actor of a right can be modified in four ways: a) by replacing the existing one, b) by adding another actor without modifying the rights of already existing ones, c) by merging the rights of the entire existing actor set, and d) by removing some actor from a right. It is also discussed that rights can be categorized into two different types, that is, use-rights and meta-rights. So for completeness, this research proposes the above mentioned four models of modifying the actor entity for use-rights as well as for meta-rights, resulting in eight different rights allocation models. The details of these models will be provided in the subsequent chapters, but their introduction is as follows:

3.2.6.1 Rights allocation

If the owner[43] agrees to give a right to another actor who agrees to take it, along with its obligations, the system should be able to allocate the right. There are four basic types of rights allocation which can be classified as:

[42] Rights allocation.

[43] As this research aims to design the models for online social interactions, the administrative authority is the owner, who is the actor having all the meta-rights over the object.

3.2.6.1.1 Replace

It allows replacing the current actor for a right over an object with a new one. After the replace operation, the previous actor cannot exercise the right but only the new actor can exercise it, for example replacing the head of department with another person.

3.2.6.1.2 Share

It allows adding a new actor to the already existing set of actors for a right over an object. After the share operation, multiple actors[44] can exercise the right without interfering each other rights and can act alone as if they owned the right exclusively, for example a couple's bank account where both can withdraw all the money.

3.2.6.1.3 Merge

It allows merging the rights of multiple actors over an object. After the merge operation, all the actors need to collaborate together in order to complete the task, for example cash withdrawal from the bank where cashier and manager both completes the request.

3.2.6.1.4 Revoke

It allows removing the existing actor from a right over an object. After the revoke operation, the actor who was previously assigned the right no longer able to exercise it. It has different characteristics and different outcomes if applied after all the above three operations, and so it is discussed under each of these operations.

Also as discussed earlier, there are two types of rights, that is, use-rights and meta-rights, it gives the table 3.1 in the context of the literature:

	Use-Rights	**Meta-Rights**
Replace	Delegation	Transfer
Share	Rights Sharing	Several Ownership
Merge	Separation of Duties	Joint Ownership

Table 3.1: Rights Allocation for use-rights and meta-rights

[44] The previous and the new.

3.3 Characteristics of rights allocation

Based on the above discussion, this research proposes the rights allocation models to deal with the modification of actor entity in the rights triplet for use-rights and meta-rights. This section now presents some characteristics for the basis of the rights allocation models. Some of these characteristics are explored in various studies in literature (Barka & Sandhu, 2000a; L. Zhang et al., 2003; X. Zhang et al., 2003) and are used here due to their helpfulness in understanding the nature of the models. The description of these characteristics is as follows:

3.3.1 Consent

This characteristic deals with whether the consent of the owner and beneficiary is required for an allocation or not. There are two possible design options for this characteristic: several consent – only the consent of the owner is required, and joint consent – the consent of both owner and beneficiary is required. The case when the allocation can be done only with the consent of the beneficiary is ignored as it would reduce the community trust on the system, which is quite important for the success of a STS.

Online scenarios where the owner can allocate the rights without the consent of the beneficiaries can be seen in giving view rights to general public over YouTube videos. The consent of both the owner and the beneficiary is required in cases when allocating rights also impose some responsibility to the beneficiary, for example assigning reviewers to papers in a conference system requires their consent. Also, by taking the consent, the owner and/or the beneficiary agree on the rules for the use of the object. These rules (allocation contract) must be met under all circumstances and failure to do so may result in revocation of rights from the beneficiary.

3.3.2 Totality

Totality of rights was first introduced in the context of machine to machine delegation, where it deals with delegating a subset of access rights from one local process to another remote process (Gasser & McDermott, 1990). In the context of user to user allocations, it becomes a characteristic that deals with allocating the proper/improper subset of rights over proper/improper subset of authorized resources (Barka, 2002; Barka & Sandhu, 2004, 2007; X. Zhang et al., 2003; Ye et al., 2005, 2006).

This characteristic refers to the amount of rights that can be allocated to another actor, and has two design options: total and partial. Total rights allocation means allocating the complete set of rights associated with an object, while partial allocation deals with allocating a subset of rights over an object. For example on Facebook, it is possible for the owner of a video to give only the view rights to friends who cannot vote or comment on it, in which case the allocation is partial, or to give all the rights over the video to friends, in which case the allocation is total[45].

In rights allocation, totality can be discussed in two ways: a) an actor can allocate a subset of his rights over all his resources, or b) an actor can allocate all of his rights over a subset of his resources. This gives four possible cases for this characteristic: a) total resources with total rights, b) partial resources with total rights, c) total resources with partial rights, and d) partial resources with partial rights.

3.3.3 Cardinality

Cardinality of an allocation refers to the number of beneficiaries in a single rights allocation operation, who can simultaneously hold an allocated right (Barka & Sandhu, 2000b; Y. Liu, Yu, & Hao, 2009; Z. Zhang et al., 2008). There are two possible design options: single-cardinal where a right over a particular object can be allocated to a single beneficiary at a time, and multiple-cardinal where a right over a particular object can be allocated to multiple beneficiaries at the same time. For example, the conference chair can assign a single person as reviewer over a research paper where the cardinality is singular, or he can assign multiple reviewers over the paper where the cardinality is multiple.

3.3.4 Monotonicity

This characteristic refers to the state of rights[46] of previous actors after rights allocation[47] (Barka, 2002; Crampton & Khambhammettu, 2006, 2008; Y. Liu et al., 2009; Z. Zhang et al., 2008). It has two design options: mutual exclusive and mutual inclusive. Mutual exclusivity

[45] Here the author is not concerned about all the possible operations on a video posted on Facebook wall, but only considered view, vote and comment for illustration purpose.

[46] How much rights remain with whom.

[47] Whether the previous actor can still exercise the right or not.

refers to the state where the previous actors do not maintain their rights over the object after the allocation and the rights of the new actor replace the rights of the previous actors over that object. On the contrary, mutual inclusivity means that the allocation makes no change in the state of rights of the previous actor, who continue to exercise it along with the new actor.

3.3.5 Depth

Depth of an allocation was first explored in (Gasser & McDermott, 1990), in the context of machine to machine delegation among processes, where it deals with the number of connections between the local machine and the remote server.

With further evolution in the context of user to machine and user to user allocations, this characteristic refers to the ability of the beneficiary after getting a right to further allocate it or not and is explored in some well-known studies (Abadi et al., 1993; L. Zhang et al., 2001, 2003; Varadharajan et al., 1991; X. Zhang et al., 2003; Z. Zhang et al., 2008). It has two possible design options: chain depth where the beneficiary can pass the right to another actor, and single pass where the beneficiary cannot pass the right to someone else. For example, purchasing a house allows the new owner to further sell it so it is chained depth, while renting a house does not allow the tenants to further sub-rent it in which case it is single pass. This characteristic is also one of the key differences between use rights and meta-rights, as allocating someone use rights does not allow them to further pass it on, whereas allocating meta-rights allow them to further pass the right to other actors.

3.3.6 Revocation

Revocation is a process by which rights are taken back from the beneficiary (Barka, 2002; Barka & Sandhu, 2000a; Crampton & Khambhammettu, 2006, 2008). Revocation of rights is of the same importance as its allocation because granting rights to actors are not static but dynamic in nature. There are three possibilities associated with revocation of a right.

a) Self-Revocation: The owner himself revokes the rights from the beneficiary based on inappropriate use of the allocated right or at will.
b) Time-based Revocation: At the time of allocating the right, the owner can assign a time-stamp with the right for the proposed lifespan of the allocation. When the time-stamp expires, the right is automatically revoked from the beneficiary.

c) Rule-based Revocation: At the time of allocating a right, a set of rules are defined for the use of right, which are known as the allocation contract. If the beneficiary violates any of those conditions, the right is revoked automatically and they cannot further exercise it.

3.4 Reduction approach

If one tries to enforce all of these characteristics with all the possible design options, the result would be quite enormous and may not be possible to cater in any practical model. So, to reduce the total number of possible combinations, this research uses socio-technical design to identify a systematic reduction approach, depicting as a tree structure, by eliminating the branches where some particular design option is not very useful. The first partitioned of the tree is based on discretionary/mandatory domains, but as online social interactions are based on ownership, the tree has overlooked the mandatory domain.

The first practically visible distinction is of monotonicity which divides the whole tree into mutual exclusive and mutual inclusive sub-branches. Further, both the sub-branches are partitioned based on cardinality, totality, consent and depth. Some of the tree branches are eliminated due to lack of interesting real world practices and the remaining branches lead to a framework, which is used as a basis of different rights allocation models for online social interactions. The reduction approach illustrated as a tree structure is shown in figure 3.3.

3.4.1 Mutual exclusive allocation

This branch deals with cases when only the owner or the beneficiary can exercise a right in a given state of the object at a particular time. It is further divided into two sub-branches with respect to cardinality: single-cardinal and multiple-cardinal.

3.4.1.1 Cardinality

In mutual exclusive side, the multiple cardinality of an allocation is not very practical in today's online environment[48], and so the multiple-cardinal sub-branch is eliminated. However, the single-cardinal, mutual exclusive sub-branch seems more useful as it deals with cases where

[48] If a right can be given to multiple actors at the same time then it can also be kept by the owner.

once a right is allocated to one actor, it cannot be allocated to another without revoking it from the first beneficiary[49]. This has many useful scenarios in STS, for example allocating the copyright of an accepted paper to only one conference. This sub-branch is further divided into two sub-branches with respect to totality: total and partial.

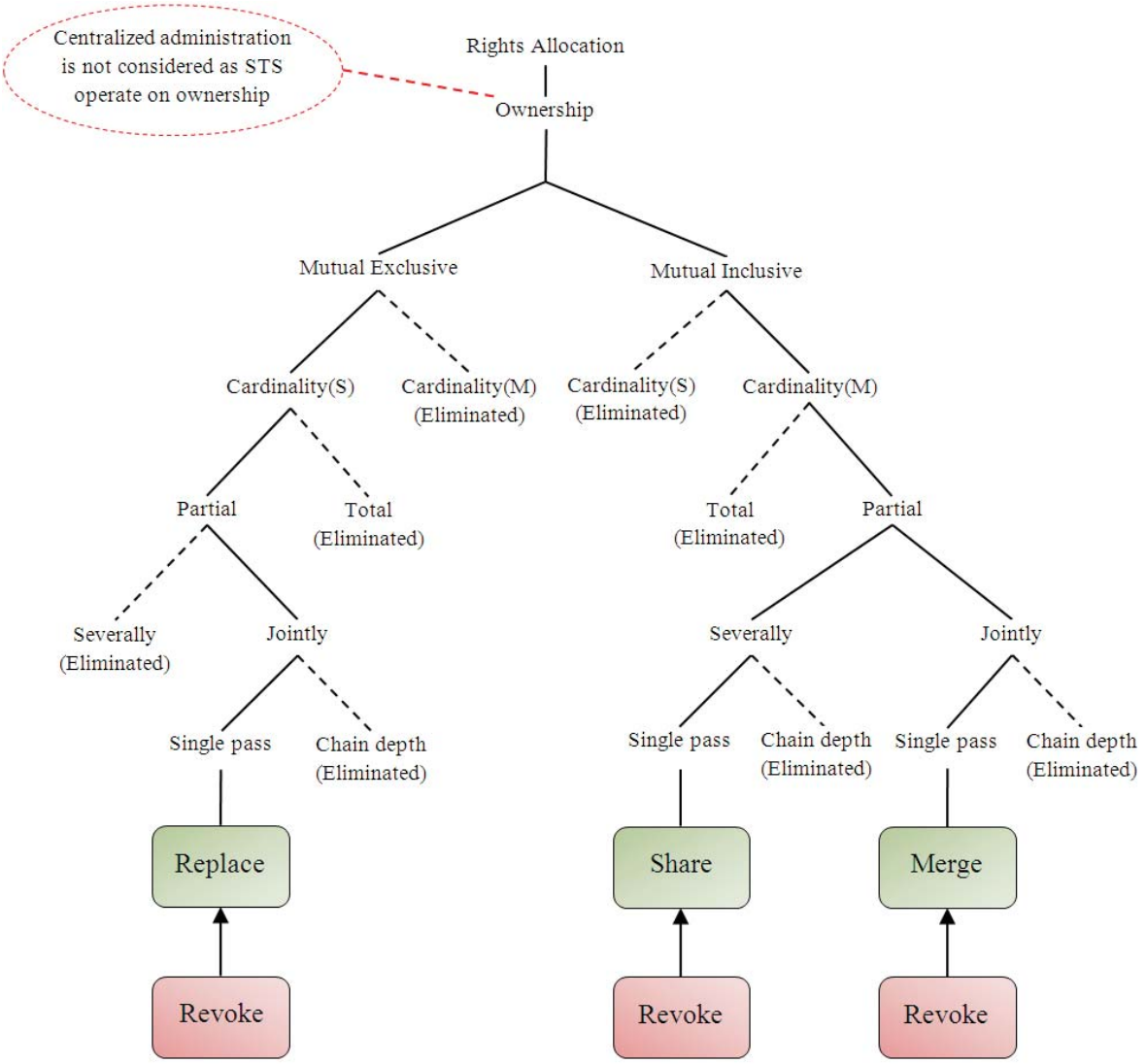

Figure 3.3: Reduction approach illustrated as a tree structure outlining various rights allocation models for online social interactions

3.4.1.2 Totality

Total allocations are less useful in current online social interactions as allocating the complete set of rights gives ultimate control to the beneficiary. Also if it is desired in some cases, it can

[49] Renting an apartment to only one tenant is a physical world example.

be achieved using multiple partial allocations. So the total sub-branch is eliminated in the reduction tree. However, the partial, single-cardinal, mutual exclusive allocation is more useful in different online social interaction scenarios as users normally give some rights to the beneficiary rather than all rights. The copyright example also supports this as it only gives the right to publish the paper but not to edit it or to remove the author's (owner) name from it. This sub-branch is further divided into two sub-branches with respect to consent: several consent and joint consent.

3.4.1.3 Consent

If rights are allocated to the beneficiary in a mutually exclusive manner, where no other actor can exercise it, the consent of the beneficiary is required. Also, the ownership principle directs that the owner's consent is always considered, so their consent is also required, so the joint consent sub-branch is taken while several-consent sub-branch is eliminated. The joint consent sub-branch is further divided into two sub-branches with respect to depth: single-pass and chain depth.

3.4.1.4 Depth

Chain depth allows the beneficiary to further pass on the rights to others, which is a key distinction between use and meta rights. It also involves the discussion of the administration of rights, so it is ignored in the framework and only the single pass option is explored[50]. So the mutual exclusive allocation branch supports the rights allocation models that work with single pass, joint consent, partial and single-cardinal allocations.

3.4.2 Mutual inclusive allocation

This branch deals with cases when both the owner and the beneficiary can exercise a right in a given state of the object at a particular time. This assumption gives many useful scenarios like YouTube videos, Wikipedia articles, and sharing of one research paper among various reviewers. It is further divided into two sub-branches with respect to cardinality: single-cardinal and multiple-cardinal.

[50] It will be discussed later, after outlining the use and meta rights models (section 7.3).

3.4.2.1 Cardinality

In mutual inclusive side, the single cardinality of an allocation is not useful at all as if a right can only be given to a single user then it cannot be mutual inclusive, so the single-cardinal sub-branch is eliminated. On the other hand, the multiple-cardinal, mutual inclusive allocation seems more useful as it deals with cases when a single right over an object is allocated to multiple actors at the same time. This sub-branch is further divided into two sub-branches with respect to totality: total and partial.

3.4.2.2 Totality

Total allocation sub-branch is less useful in online social interactions as allocating the complete set of rights gives ultimate control to the beneficiary, and having multiple-cardinality makes it even worse. So the total, multiple-cardinal, mutual inclusive sub-branch is eliminated in the reduction tree. However, the partial, multiple-cardinal, mutual inclusive sub-branch is more useful in different online social interaction scenarios as users normally give some rights to the beneficiary rather than all rights. The view right example of YouTube video also supports this as it only gives the view right over videos to the general public but not the right to edit or delete it. This sub-branch is further divided into two sub-branches with respect to consent: several consent and joint consent.

3.4.2.3 Consent

Both of the options in this sub-branch are often used in current online social interaction scenarios, like viewing the video not requires the joint consent while accepting a paper at some conference requires the joint consent of all the reviewers. These practical opportunities lead this research to take both of these sub-braches into account for rights allocation framework for online social interactions. Both of these sub-branches are further divided into two sub-branches with respect to depth: single-pass and chain depth.

3.4.2.4 Depth

Again, as the depth of an allocation relates to the discussion of meta-rights, the chain depth option, where the beneficiary can further pass on the right, is ignored. This gives that both the mutual inclusive sub-branches operate on single pass.

The above discussion provides two different mutual inclusive sub-branches, where the rights model should support single pass, several and joint consent, partial and multiple-cardinal rights allocations. This gives three major rights allocation models (Replace, Share and Merge) as illustrated in figure 3.3.

3.5 Chapter summary

This chapter has outlined the basic components for this research. It has explored the ownership and relationship properties of online social interactions from the literature, and has discussed some basic access control components and their semantics in the STS environment. It has clearly distinguished two types of rights and four basic types of operations that are possible on those rights, to illustrate the scope of this research. It has further explored various characteristics of rights allocation and has outlined a framework for online social interactions based on socio-technical design. This framework will be used in the later chapters to define the models.

Chapter 4

Social Access Control Model

This chapter outlines the basic access control model for online social interactions based on the properties explored in the previous chapter. This social access control (SAC) model needs to support ownership, local administration, relationships and object classification. This chapter is organized as follows: First the ownership framework and its semantics have been outlined along with role assignments and initial space configurations. Further various components of the model are presented followed by its logical definition and the grant process. This social access control model will be later used as a basis for various rights allocation models.

4.1 Ownership framework

To create an information object from nothing is as impossible in an online space as it is in a physical one. Creation cannot be an act upon the object created, as it by definition does not exist before it is created. Likewise, an actor cannot request an access control permission to create for an object that does not exist yet. Also, to create an information object its attribute structure must already be known, that is, exist within the system. To be consistent, creation is an act upon the system, or in general, an act on the space containing the created object. This gives the design principle:

P.4.1[51]*: Creation is an act on a space, up to the system space.*

It is well defined as a system always has some space, and the system itself is the first space. Creating is also an act upon a space because it changes the space, as it now contains the created object. If creation is always an act upon a space, it follows that the right to create in a space belongs to the space owner:

$$Right_{Create}(Owner_{space}, Space, Operation_{create}) \dots [eq.4.1]$$

where *Space* is the object, $Owner_{Space}$ is the actor who has created the space, while $Operation_{Create}$ is the operation performed on that space.

This allows an access control system to be initialized with a system administrator (SA) owning a system space with all rights, including create rights, which then evolves into a community as the *SA* gives rights away. To create a community of others, one must give rights away (Gaaloul et al., 2008).

The logic can generalize to any space – the right to create in the space is initially allocated to the space owner who can allocate it to others who enter the space. For instance, to create a board post, YouTube video, blog comment or conference paper requires the board, video, blog or conference owner's permission.

4.1.1 Role assignment

If every object is created in a parent space and can contain other objects as its children, it requires some generic roles associated with every space. Following are the roles that can be assigned when an entity is created:

4.1.1.1 Owner

Owner role is associated with every object and has all the use-rights and meta-rights over the object. This conveniently resolves the issue of how to allocate the rights to newly created object – they are allocated to its creator, including meta-rights. Create then immediately gives the right to edit which is useful as create sets no new object values. Yet it is not what must happen – a technical program can create an information object however it likes, for example

[51] Should be read as P(Principle). 4(Chapter number). 1(Principle sequence number within the chapter).

give its ownership to the system administrator as in traditional applications. Creator ownership is a requirement for social success not a technical necessity. This gives the access control design principle:

P.4.2: The creator of new entity should immediately gain all rights to it.

As the owner role gets all the rights over the object, it gives

$$Role_{Owner} = (Owner, Object, Operation_{all}) \ldots [eq.4.2]$$

where *Object* is the object that is created in a space, *Owner* is the actor creating the object, while $Operation_{all}$ is the grant for all operations that can be performed on that object.

4.1.1.2 Parent

The parent role is assigned to the owner of the parent space where the object is created[52]. As a created entity becomes part of the space it is created in, it should be visible to its parent space owner who is accountable for their space[53]. By the same logic, an entity should be visible to all its parents, giving the design principle:

P.4.3: A space owner should have the right to view/delete any offspring.

It gives:

$$Role_{Parent} = (Owner_{space}, Object, view\ /\ delete) \ldots [eq.4.3]$$

where *Object* is the object that is created in the space, $Owner_{Space}$ is the actor created the space, while *view/delete* are the operations granted for the role. A posted conference paper could be visible to its conference, track and mini-track chairs, but not to other track or mini-track chairs. Parents may receive notifications of new additions.

4.1.1.3. Offspring

The offspring role is assigned to the owner of the child object in perspective of the owner of the space. As the child has already entered its parent space, they can view whatever is displayed

[52] It is a role from the perspective of the owner of the object.

[53] Accountability also allows the parent to delete the objects (or restrict its visibility) created in their space.

in it, that is, child can always see the space they are in. By extension, they can also enter any parent space as they are already in it, giving the access control design principle:

P.4.4: An entity owner has the right to enter its parent.

For example, adding a paper to a mini-track should let one enter the mini-track, track and conference spaces to view whatever is displayed there. The offspring role is:

$$Role_{Offspring} = (Owner_{offspring}, Object, Enter) \ldots [eq. 4.4]$$

where *Object* is the object that is created in the space, *Owner* $_{Offspring}$ is the actor created the child object, while *Enter* is the operation granted for the role.

4.1.1.4. General public

A space owner can create a local public role, to define what others can see or do in the space:

$$Role_{LocalPublic} = (LocalPublic, Space, Operation_{any}) \ldots [eq. 4.5]$$

where *LocalPublic* is the actor visiting the space, while *Operation* $_{any}$ is any operation granted for the role by the space owner. Actors in a local public role can be set manually by the owner, as friends are assigned, or set to a general public list given by the system. A space owner can grant any right they own to their local public subject to conditions, for example Wikipedia create condition is to allow public edits.

A complete list of rights[54] for various roles associated with the created object is given in table 4.1.

Parent	Owner	Offspring	G. Public
UR(V, D)	UR(V,D,E) MR(V,D,E)	UR(V)	D/C
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively D/C represents don't care and depends on space configuration for G. Public role			

Table 4.1: Right for different roles associated with the created object

[54] This dissertation has taken these rights as the general common subset for most applications, see CRUD (Create, Read, Update, Delete) description in the context of software engineering.

4.1.2 Space initial configuration

These conditions are applied when a space owner assigns various rights in his space to different default role set, where they can limit:

a) *Object type*. The space owner may limit the object type created, for example in a conference, the right to create paper in a track is not the right to create a mini-track.
b) *Operations*. The operations allowed on created objects, for example a comment is not usually editable once posted but ArXiv lets authors edit publications.
c) *Exclusivity*. Who can access created objects, for example journals give authors exclusive edit rights while Wikipedia lets anyone edit any creation.
d) *Visibility*. Who can view created objects, for example bulletin boards let you see what others submit but conferences do not until the review phase is done.
e) *Defaults*. Space owners set created entity default values.
f) *Rights of Roles*: The default set of rights associated with each role, for example YouTube allows general public role to view the video, while Wikipedia allows them to edit the articles.

A space owner can assign creation rights as needed, for example to set vote results to only show to people who have voted, to avoid bias.

4.2 YouTube create process demonstration

Technically, creating an entity is simple – a program just creates it, but socially adding to another's space is not a one-step act, for example adding a YouTube video involves:

a) *Registration*. Creating a YouTube public role persona.
b) *Entry*. YouTube allows public entry, if not banned.
c) *Creation*. YouTube lets the public role upload videos.
d) *Edit*. One gets edit right to title, notes and properties.
e) *Submit*. To submit it to the public view.
f) *Display*. The space displays it so the public sees it.

In this model, YouTube gives create video rights to anyone who has registered in the public role (a). They enter the YouTube space (b) and create a video by uploading or recording,

which they own (c). They can then in private view it and edit details (d). At this point, the video is visible to themselves and administrators but not to the public, and they can still delete it. The video is then submitted to YouTube for display to its public (e), which usually occurs quickly as YouTube gives display rights (f). Note that create, edit and display a video are distinct steps. As YouTube only gives display rights, it can still reject videos that fail its copyright or decency rules by taking the rights back. This rejection is not a delete, as the video owner can still view, edit and resubmit it.

In contrast, a purely technology based system might let space owners delete items at will. Ignoring creator ownership would discourage social participation and the system might fail socially. The above logic generalizes easily, for example a YouTube video is itself a space with dependent comments and votes. YouTube is consistent and fair as the same principles apply when the video creator becomes a space owner. They can choose to allow comments or votes on their video, that is, they can grant rights to their domain space, just as Facebook citizens do. STS succeed by allocating social rights legitimately (Whitworth & deMoor, 2002, 2003).

4.3 Social access control model

From the previous discussion, it is clear that access control for STS should support the basic requirements of ownership, dynamic asymmetric relationships, local administration, local visibility and object classification. This section now outlines the distributed social access control model for STS based on the above mentioned concepts.

The ownership resolves the issue of who will manage the rights over online objects. By extension, it also resolves the issue of the ownership of the personae. In the physical world, freedom is the right to control one's body, to not be a slave to another. If freedom online is the right to control one's online body, or persona, one should be able to edit or delete it.

Further local visibility demands to restrict the visibility of objects to some virtual domain. All the objects by one owner can be collected in a domain, and categorized according to their privacy level. These domains can be locally administered by their owners who are responsible for managing the rights of other users over their objects. Relationships can also be managed through domain based roles that are administered by the owners. These domain based local roles provide fine-grained control over resources and their access. To make it asymmetric, two

unidirectional information objects can be stored in each domain accessible and modifiable by each owner. The introduction of unidirectional relationship objects also solves the problem of heterogeneous individual traversal policies and allows every user to formulate their policy according to their own discretion.

4.3.1 Definitions

Table 4.2 defines the constructs of the social access control model.

	Definition
SH	*Stakeholder:* The owner who posts system resource objects, for example photos, videos, comments or votes.
NS	*Namespace:* The set of objects a stakeholder creates.
VU	*Virtual user:* A user, from the social circle of stakeholder, seeking a NS resource access.
LR	*Local role:* A VU group with defined access to NS resources.
OC	*Object class:* An object group, based on security clearance, whose access is mapped to LRs.
AC	*Attestation certificates:* Permission objects encapsulated various access rights and map LR to OC.

Table 4.2: Abbreviations and their definitions

4.3.2 Components

The concept of stakeholder is introduced to support ownership, namespaces are introduced to support local administration, virtual users and local roles are introduced to support fine grained relationships, object classes are introduced to support object privacy classifications, while attestation certificates are introduced to support asymmetric dynamic relationships. The details of these components are as follows:

4.3.2.1 Namespace

Every STS allows its users to administer their resources and local roles associated with them. Also the role management is done by the users themselves. For instance, if *Alice* belongs to the friend group of *Bob*, it is the responsibility of *Bob* to manage membership of *Alice* to his friend role as well as administer rights for friend role over his resources. This divides the whole STS into multiple administrative domains managed by multiple users (owners).

The proposed model distinguishes between owner of a resource and the virtual users accessing them, and divides the system with respect to the owners owning resources. The whole system is divided into multiple namespaces with owner administering his namespace along with the resources in it. Dividing the system into autonomous namespaces reduces administration and processing costs, and enhances user trust. It lets stakeholders control their own domain. They can classify virtual users by relationship, and objects by who can see them. Mapping local roles to object classes manages access control.

4.3.2.2 Local roles

Roles like parent, friend or boss are used in norms and laws to simplify rights management, for example owner as the generic party with the right to use an owned object. In online access control, roles both simplify rights management and improve social acceptability. Roles are loosely seen as an actor set, but here are an actor set in a rights statement, for example the friend role is a set of people in the context of stated permissions. So roles are here generic rights, giving the access control principle:

P.4.5: A role is an entity right expressed in general terms.

The model allows local roles to restrict access to resources. Roles represent classes of permissions controlled by their namespace. A virtual user seeking access to a resource must acquire a role in the namespace with the requested access rights. Local roles have namespace wide scope and do not exist beyond that. They are dynamic, as different stakeholders can implement different roles and one virtual user can concurrently acquire various roles in multiple namespaces.

Roles reduce rights management complexity and are flexible enough to accommodate social variety, for example the friend role lets one add or remove others who can view photos posted on a social network wall:

$$Role_{Friend} = (Actor, Entity_{wall}, Operation_{view}) \ldots [eq.\ 4.6]$$

where *actor* is the entity having *view* right over *wall* by getting the *friend* role. To friend another adds them to a role actor set and to unfriend removes them. To 'friend' does not change the target persona but the actor's role, so it is really an act upon a role.

4.3.2.3 Object classes

In this model, objects present in one namespace are classified according to their privacy clearance. They are first put in an object class, or container, and then the class is given a privacy clearance depending on its objects. This clearance is based on the contents of the objects depending on the owner discretion[55]. Access mappings are between object classes and local roles, not between objects and actors.

Local roles and object classifications give owners both the simplicity of high control and the flexibility of low level control. In high level control, users link abstract roles to abstract object classes that make user sense, for example whether to let friends view the family photo album. In low level control, users can define a new role of one person or a new object class of one object, to define exactly who can see what, allowing a degree of fine grained access control not currently offered. This introduces new communicational aspects and helps to enhance social relationships.

4.3.2.4 Attestation certificates

To introduce an additional security layer between *OC* and *LR*, the concept of attestation certificates *(AC)* is introduced similar to (Thompson et al., 1999). These client-side distributed certificates are local to *NS* and helpful in maintaining unidirectional relationships. For example, when *Alice* establishes a relationship with *Bob*, she creates a local certificate stating that *Bob* is a colleague of *Alice*. This certificate is locally stored in her *NS*, accessible to her only, so cannot be forged. A similar but independent certificate is generated and stored by *Bob* as well, where he may consider her as friend. When *Bob* requests a resource from her domain, *Alice's* access control module matches *Bob's* 'userid' against her roles to decide the request outcome. Having one way attestation certificates provides flexibility and also resolves the problem of certificate forgery. In contrast, centralized certificates need more processing and also expose a single point of failure. Further it is liable to security attacks, in that the resource owner and the requestor both have certificates in the same repository and can access and modify them.

[55] One owner can tag a class as public while the other can tag a similar class as private.

4.3.3 System architecture

These features define an access control model independent of the policy. Each *SH* manages its access control policy by allocating *VU*s to *LR*s with known access to *OC*s. No global administration is needed, as *SH*s administer their *NS* resources by mapping *LR*s to *OC*s.

The *VU* are not directly mapped to the resources rather the entry point to a *NS* is the abstraction of roles. All the *VU* in *SH NS* are assigned some *LR* and access to objects is granted on the basis of *LR* membership. Also, the objects *O* in *SH NS* are categorized into some security labeled *OC* with respect to their disclosure level. Every *LR* is assigned an *AC* and the access decision is made on the encapsulation of requested right in *AC* for the requested *OC* label. The system architecture of social access control model is illustrated in figure 4.1.

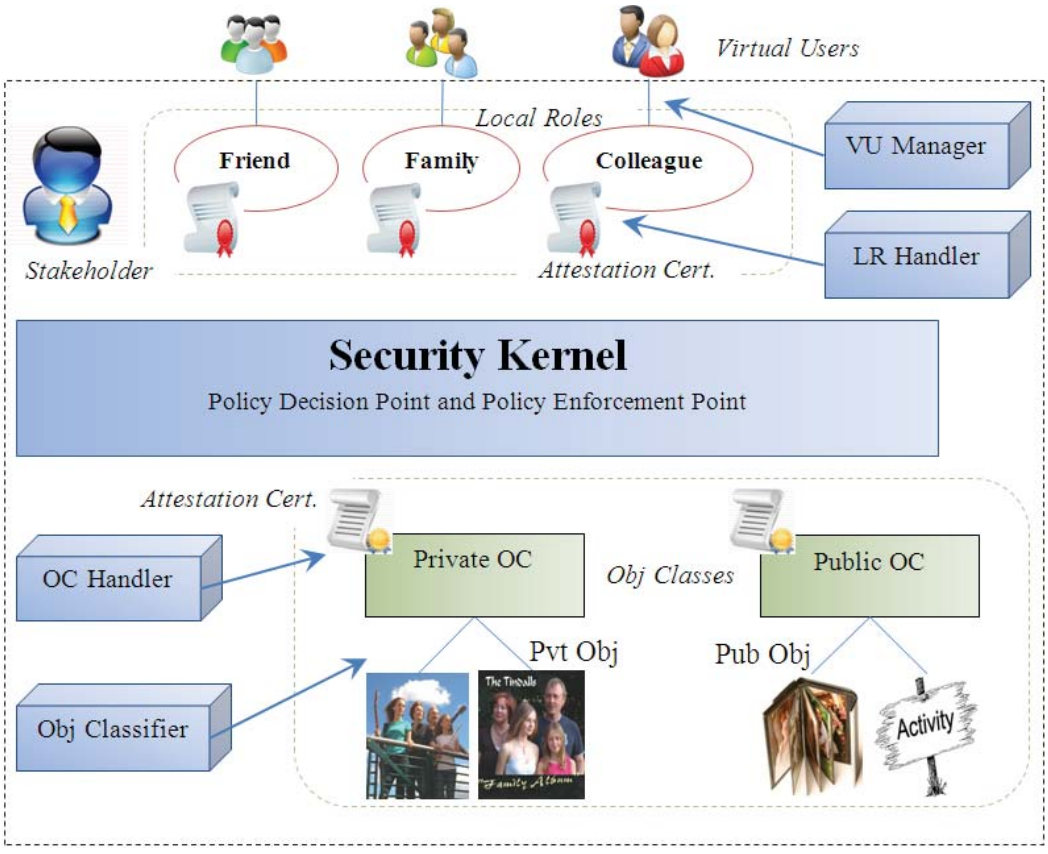

Figure 4.1: Distributed access control model system architecture

An access control policy is a definition of how a system should provide or deny access which can range from an abstract statement like, "only users on this list should have access", to

policy languages with executable (operational) semantics (Chu, Feigenbaum, Lamacchia, Resnick, & Strauss, 1997; Li, Grosof, & Feigenbaum, 2003). As described earlier, STS users do not have the same global access control policy working for everyone; rather it is different from everybody else depending on their social structure. One may want to share all her information with everyone, whereas one may want to restrict it to her social circle only.

4.3.4 Definition

Based on the above components, the SAC model can be defined as follows:

SH, *NS*, *VU*, *LR*, *OC* and *AC* are sets of Stakeholders, Namespaces, Virtual Users, Local Roles, Object Classes and Attestation Certificates respectively.

- The system space is divided into different administrative namespaces denoted by *NS*.

$$NS = \{ns \in Space_{system} | \bigcup_{i=1}^{N} \{ns_i\} = Space_{system}\}$$

- Objects associated with a single owner are gathered in a single namespace resulting in dividing the whole system in autonomous *NS* owned by *SH*. The stakeholder *SH* is the trusted authority to perform administration of a namespace.

$$SH = \{sh \in Users | (sh, ns) \rightarrow admin_{ns} \wedge \bigcup_{i=1}^{N} \{sh_i\} = Users\}$$

where *Users* is the set of all users present in the system and $admin_{ns}$ is the administrator of the namespace.

- *VU* denotes the set of virtual users present in *SH* social circle.

$$VU = \{vu \in ns | \bigcup_{i=1}^{N} \{vu_i\} = SocialCircle_{sh}\}$$

where *N* is the number of *VU* present in a NS_i

- *LR* denotes the set of local roles present in *SH NS*. *LR* has local scope in *SH NS*, and *VU* are associated with them to access the resources. A many-to-many virtual user to local role assignment relation is given by

$$VU\_LR \subseteq VU \times LR$$

- $VU\_assign: (lr{:}LR) \rightarrow 2^{vu}$ is a function derived from VU_LR for mapping of local role lr onto a set of virtual user in a namespace.

$$VU\_assign(lr) = \{lr \in LR, vu \in VU | (lr, vu) \in VU\_LR, \forall\ LR, VU \in NS_i\}$$

- Every object in the NS is categorized into object privacy classes, denoted by OC, under some label (default $L_1(\tau)$), that is, $O \rightarrow OC_\tau$. Where, τ is the set of all security labels that are used for confidentiality levels and the access to object is managed by the OC label τ it contains. These labels are hierarchical under a partial order $\succ$ such that $L_1 \succ L_2$ if and only if $L_2 \in L_1$. A many-to-many object to object class assignment relation is given by

$$O\_OC \subseteq O \times OC_\tau$$

- $Obj\_assign: (oc{:}OC) \rightarrow 2^O$ is a function derived from O_OC for mapping of object class to a set of objects in a namespace.

$$Obj\_assign(oc) = \{O_i \in O, OC_\tau^i \in OC_\tau | (O_i, OC_\tau^i) \in O\_OC, \forall\ O, OC \in NS_i \wedge \tau \in SH_{Policy}\}$$

which assigns the OC_τ to objects present in the NS under the security label τ depending on the policy of namespace stakeholder.

4.3.5 The access control process

This section illustrates some basic default conditions that a VU must satisfy to get access to a resource. These conditions (as a subset of policy) are stored in the NS that only the SH has access rights to. The access control model defines the access control operation and grant access to a virtual user over a resource using the following rules:

- The requestor VU_j is mapped to some LR_i in the NS_i.

$$vu_j \rightarrow LR_i,\ \forall\ LR_i \in NS_i \ldots [eq.4.7]$$

- The requested object O is mapped to some OC_τ.

$$O \rightarrow OC_\tau, \forall\ OC_\tau \in NS_i \ldots [eq.4.8]$$

- LR_i has been granted attestation certificate which encapsulates the requested right over the requested object class.

$$LR_{AC} \rightarrow OC_{\tau}, \forall\ LR_{AC}, AC, OC_{\tau} \in NS_i \ldots [eq.4.9]$$

Given a *NS i*, an access request is only granted if the virtual user VU_j acquires a local role LR_i in namespace NS_i. Also the object *O* needs to be mapped in object class OC_τ with the privacy label τ. Further the LR_i has a valid attestation certificate *AC* that has the mapping to the requested OC_τ in the NS_i.

4.3.6 Theoretical assessment

In section 3.1.5, this dissertation has discussed how the concept of restricting the visibility of objects to the social circle of owner is quite useful in reducing the complexity of traditional access control models. As objects are not mapped to the whole user set of the system rather to only a small subset of it, the visibility of resource reduces to local roles. This concept of local visibility is used in the many access control models of STS to reduce the complexity, which would be enormous as discussed earlier.

The proposed model refines the visibility from social circles to domain based local roles, introduced the object classes based on their privacy clearance and encapsulated various rights over object classes into one attestation certificate. The authorization matrix is distributed among various stakeholders and for one namespace it is reduces to

$$VU \times LR \qquad Object \times OC \qquad Rights \times AC \qquad LR \times OC \times AC \ldots [eq.4.10]$$

and the sum of the entire authorization matrices for the whole system under the proposed model is

$$\sum_{i=1}^{N} \left[\sum_{j=1}^{x} LR_j \times \sum_{k=1}^{y} OC_k \times \sum_{l=1}^{z} AC_l \right]_i \ldots [eq.4.11]$$

where *N* is number of users present in the whole system, *x, y* and *z* are the number of *LR, OC* and *AC* present in particular *NS*, *LR* and *OC* represents local roles and object classes present in the *NS* respectively, and *AC* represents attestation certificates used for mapping of *LR* and *OC* in the *NS*.

These settings under the proposed model give fewer access control entries. For the above case, the number of entries is reduced from 33 trillion to 3.7 trillion. Figure 4.2 shows the number of access control entries generated by user number for a fixed object contribution in the last two cases – current existing access control models for social networks and the proposed model.

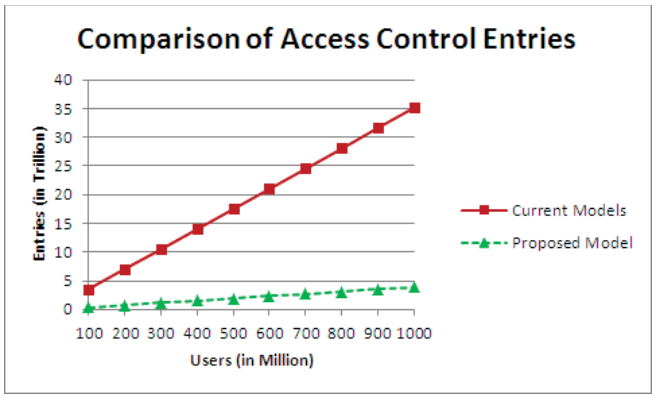

Figure 4.2: Access control matrix magnitude for different models based on [eq. 3.4] and [eq. 4.11]

4.4 Improvements over previous models

The social access control (SAC) model has contributed to the literature in the following manners:

a) The social validity of the SAC model comes from its support to the social principles of
   i. Ownership by stakeholder namespace
   ii. Freedom by giving stakeholder right over persona and its updation/deletion
   iii. Privacy by allowing heterogeneous privacy policies by mapping various virtual users to local roles, and local roles to object classes
   iv. Dynamic asymmetric relationships by introducing one way locally stored attestation certificates
   v. Object visibility by reducing it to the local roles present in the namespace
   vi. Object classification by allowing the owner to group objects with similar privacy disclosure level.

b) The concept of domains is used in STS without any formal semantics based on local requirements and programmer's intuition, so there are many variations of it. This model proposes the owner oriented domains for STS to support ownership and local

administration. It defines various components of the domain and explores their interaction.

c) Fine grained social circles were proposed in (J. Li et al., 2009; Tapiador et al., 2011), but they do not support the dynamic nature of friendship. This model not only provides full control to the owner over modification of relationship, but also gives the notion of fine grained social circles.

d) The object tagging was explored in (Hart et al., 2007), but it is based on rules rather than owner preferences and thus has less accuracy. This model has introduced owner oriented object privacy classes to support object classification based on their contents and owner privacy policy.

e) It has introduced attestation certificates in STS, which also contributed towards decentralized access control credential distribution. These certificates provide a mechanism to support asymmetric relationships.

f) It has introduced initial space configuration to support different types of online applications using the basic social access control model. These configurations allow the SAC model to be used for most types of online applications supporting social interactions.

The SAC model refines and extends the basic components required for online social interactions. The refined model supports dynamic asymmetric relationships, owner oriented object privacy classes, fine grained social circles, and ownership domains. These basic components are further used it as a basis of rights allocation models for online social interactions presented in chapter 5 and 6.

4.5 Summary

This chapter has outlined the social access control model for STS based on the requirements of creator ownership, relationship and object classification. It has given the ownership framework in a hierarchical structure, where objects are created within spaces. Various roles associated with each object are defined to describe some basic rules to initialize and configure the particular type of STS. The roles are generic but the rights associated with them can be modified according to the application requirements to provide a generalized framework. This chapter has further defined the constructs of the social access control model followed by their

description. These construct give the basis of a generalized model for most type of STS based on ownership and local administration. Also, the logical model is defined and the access control process is provided to show the semantics of the proposed model. Further, theoretical assessment of the proposed model is done to show that it does not reduce the efficiency rather it increase by some extend. The chapter then concludes by outlining the improvements of the model.

Chapter 5

Use-rights Model

Most of the studies in literature have used different terms for use-rights models and meta-rights models. This includes separate models for rights delegation and rights transfer (Barka, 2002), separate models for RBAC (Sandhu et al., 1996) and their administration (Munawer, 2000), different models for usage control and its administration (Park, 2003), delegation and transfer models for RBAC (Crampton and Khambhammettu, 2008) and many others (Dewan & Shen, 1998; Indratmo & Vassileva, 2007; L. Zhang et al., 2001, 2003; Mattas et al., 2006; X. Zhang et al., 2003). This research has outlined the four general operations possible on objects and rights, and will discuss their similarities and differences in the subsequent chapters. The issue, that use-rights and meta-rights are different but their allocation models can be same, will be discussed in detail in chapter 7, along with the possibility of designing a generalized model for both of them.

This chapter establishes the basic understanding behind the allocation of use-rights in online social interactions. Use-rights are system permissions for actors to apply operations on objects (Whitworth & deMoor, 2002, 2003). As discussed in chapter 3, there are four possible options to allocate use-rights and this chapter outlines these models in detail. The models introduced are Replace $_{Use}$, Share $_{Use}$, Merge $_{Use}$ and Revoke $_{Use}$. However, as the revoke process is different for each of the three allocation models, it is discussed under each allocation model section rather than in a separate revoke model section. Every model is described by using four distinct

steps: a) the general description of the model along with various rights allocation characteristics and their values for online social interactions, b) the mapping of various components of SAC model to describe the working of the model in an online social interaction instance, c) the logical definition of the model to illustrate the exact possible combinations of rights over an object for the roles of owner and the beneficiaries, and d) the rights analysis of different roles of parent, offspring and general public associated with the object. After outlining each allocation model, its revocation process is discussed.

5.1 Replace $_{Use}$ model

Replace $_{Use}$ model allows the owner to replace the actor for use-rights, and is commonly termed as delegation[56] (Abadi et al., 1993; Gasser & McDermott, 1990; Varadharajan et al., 1991). This is done by replacing the actor entity in the use-rights triplet, but the meta-rights triplet remains the same. The need of Replace $_{Use}$ arises when the owner of an object wants to delegate the use-rights over his object to another actor who can exercise the rights on the owner's behalf (Barka & Sandhu, 2004). After the delegation operation, the owner acquires the role of delegator – who cannot exercise the delegated right anymore, while the beneficiary becomes the delegatee – the only responsible actor for the delegated right as all the other actors holding that right are also replaced by him (Barka & Sandhu, 2000a). Delegating a right changes the actor for use-rights but not for entity meta-rights, so it can be revoked by the owner, for example after lending a book to a friend it can be taken back at any time. Delegation is used in scenarios when one actor assigns some task to another, and wants him to act independently to carry out that task even without the interference from the owner (Gaaloul et al., 2008). In the physical world, renting an apartment to a tenant, lending a book to a friend, and assigning an acting head of department are some of the examples of delegation. In these examples, the tenant/borrower/acting head can use the rights for a specific amount of time without any interference.

Let's consider the scenario of a conference system, which enables knowledge sharing and collaboration through academic peer reviews (Whitworth & Friedman, 2009a). Online

[56] As this research focuses on actors involved in online social interactions, every allocation should be considered as user to user right allocation, unless specified otherwise.

conference systems let researchers upload their work and assign various academic peers as reviewers to assure the quality of the submitted work. The system can run multiple conferences simultaneously, each with different fields, qualities, rules and other properties associated with it (Whitworth & Friedman, 2009b). Every conference consists of an independent space with various tracks[57] in it, and different roles[58] like organizers, publishers, reviewers, authors, attendees and so on, are associated with it. For big conferences where one conference chair is not able to handle all the tracks within the conference, it is normal to assign different track chairs[59] to different tracks. The system administrator is the owner of the system space, the conference chair is the owner of the conference space, while each paper/review/comment submitted to the conferences has a different owner associated with it.

To map this scenario on the core SAC model, the conference system is the parent space for all the conferences owned by the system administrator. The conference space is the namespace with conference chair as the stakeholder. Different tracks are the object classes with papers/comments/vote as objects, and organizers, publishers as the local roles associated with each conference. Various virtual users are assigned to these local roles and permissions over objects are assigned to local roles through attestation certificates.

Now, suppose that *Alice* – the conference chair for a *'Security Conference'* wants to give *Bob* all the use-rights over the *'Access Control'* track and assigns him the role of track chair. She wants *Bob* to have full authority over the track for the entire duration of the conference without any interference, so he can work freely. He can create and manage mini-tracks, assign reviewers, accept/reject papers, and is fully accountable for the success/failure of his track. At the same time, *Alice* also wants to maintain her authority over the whole conference and tracks, so if *Bob* is unable to manage it properly she can take it back. In this case, she replaces the use-rights actor for the track *'Access Control'* with *Bob*, who can act freely over the whole track. This scenario is depicted in figure 5.1.

[57] For objects (research papers) of different types (areas).

[58] For actors having different privileges.

[59] Who are considered as the actual responsible person for that track.

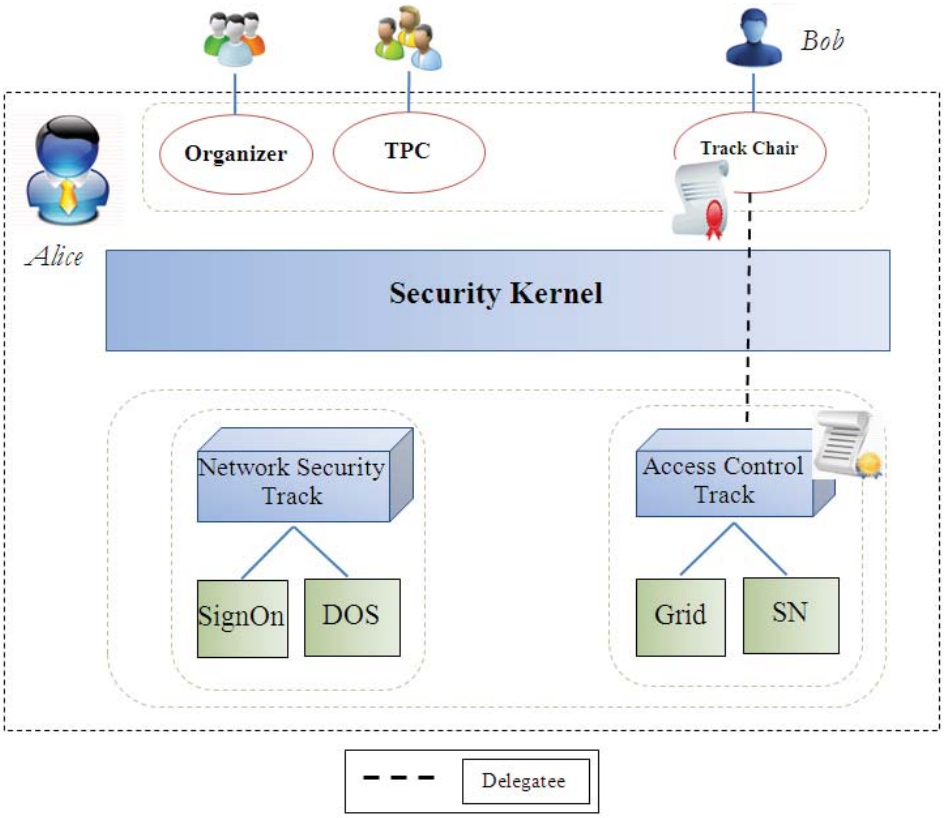

Figure 5.1: Replace $_{Use}$ scenario depicting a running conference

5.1.1 Characteristics of Replace $_{Use}$

Following are some of the characteristics of this Replace $_{Use}$ model that distinguish it from other models. Some of these characteristics are explored in literature (Barka & Sandhu, 2000b; L. Zhang et al., 2003; X. Zhang et al., 2003) for role to role delegation in RBAC (Ferraiolo & Kuhn, 1992; Ferraiolo et al., 2001; Sandhu et al., 1996). However, the values of these parameters are different in this research from the values proposed in the literature due to the nature of online social interactions and the ownership domain.

There are two basic axioms for this model, which are also kept consistent throughout all the other use-rights models defined in this research. First, the Replace $_{Use}$ model presented in this research is domain based, as rights are associated with objects, and second, no one can share a use-right associated with an object which they do not own. So, *Alice* can only delegate a right to *Bob* if she holds the meta-right for that right over the object. It is not possible for her to assign *Bob* rights over another conference for example *'Trust Conference'*, where she does not have any meta-right.

5.1.1.1 Consent

As success/failure of tracks counts towards success/failure of the whole conference, and as the delegatee will work on *Alice's* behalf, her consent is required before *Bob* has delegation of any responsibility. If *Bob* can get rights over a track without *Alice's* consent, she will not trust the system and would not use it. Also, running a track means that *Bob* is responsible for the success of his track, which includes attracting the researchers, assigning the reviewers and incorporating the results. All of these tasks require time and effort, so *Bob's* approval is needed before he can be assigned the role of track chair.

The above implies that delegating a right means that the delegator is agreeing to trust the delegatee with the delegated rights, and delegatee is agreeing to take all the responsibilities associated with the object. This is important as an online system may not succeed if it modifies rights without the consent of the owner. It gives the design principle:

P.5.1.1[60]*: Consent of the delegator and the delegatee is required for every delegation.*

5.1.1.2 Totality

Totality deals with allocating the proper/improper subset of rights over proper/improper subset of authorized resources. In the above example, if *Alice* has all the rights over the object, it is up to her that how much she wants to delegate to *Bob* from her set of rights. She may want to delegate all the use-rights[61] over the track[62] to *Bob*, in which case the delegation is total, or she may want to assign *Bob* as track chair – assigning only a subset of her complete set of rights[63], but keeping the rest[64] with her, in which case the delegation is partial. This model supports allocating the partial rights over partial set of resources. Total delegation reduces the efficiency as only one beneficiary can work on one object, which does not scale well over multiple domains and cannot support task specialization.

---

[60] Should be read as P(Principle). 5(Chapter number). 1(Model sequence number within the chapter). 1(Principle sequence number within the model).

[61] View, edit and delete, for example.

[62] As the illustrated scenario discusses a single track, it is implied that the model supports delegating partial set of resources.

[63] View and edit.

[64] Delete.

Generalizing the above implies that the Replace $_{Use}$ model can operate on two design options, that is, only complete delegation is supported so all the use-rights or none can be delegated, or partial delegation is supported so part of use-rights can be delegated. This model supports partial delegations as it is more useful and also total delegation can be achieved using multiple-partial delegations, which gives

$$Rep_u(UR_{all}) = Rep_u(UR_1) \wedge Rep_u(UR_2) \wedge \ldots \wedge Rep_u(UR_n) \ldots [eq.5.1]$$

This gives the design principle:

P.5.1.2: The owner can delegate proper/improper subset of their rights.

5.1.1.3 Cardinality

Cardinality deals with the number of beneficiaries in a particular rights allocation operation. In the running scenario, *Alice* can delegate management responsibilities over the *'Access Control'* Track to *Bob* and financial responsibilities over the same track to *Carl*, which gives that she can delegate different rights over the same object to multiple beneficiaries. This gives

$$Rep_u \rightarrow (UR_1)\{Bob \mid Bob \in VU \; \forall \, VU \in NS_i\} \ldots [eq.5.2]$$

$$Rep_u \rightarrow (UR_2)\{Carl \mid Carl \in VU \; \forall \, VU \in NS_i\} \ldots [eq.5.3]$$

However, she cannot delegate financial responsibilities to *Bob* and *Carl* at the same time, resulting in conflict of delegated rights[65]. The conflict of rights occurs when she delegates some rights to *Bob* and the subset of *Bob's* rights to *Carl*, where rights sets are mutually inclusive. So, the case

$$Rep_u \rightarrow (UR_3)\{(Bob, Carl) \mid Bob, Carl \in VU \; \forall \, VU \in NS_i\} \ldots [eq.5.4]$$

will raise conflict in the system and is not allowed in the model.

Generalizing the above scenario gives that the rights over an object can be delegated to multiple delegatees, but their rights should not conflict. The support for cardinality is

[65] If it is desired in some case, it can be achieved by applying Replace $_{Use}$ first and then applying Share $_{Use}$ (Section 5.2).

important because one namespace can be delegated among many actors resulting in separation of duties, and avoiding the conflict is important for accountability reasons[66]. This gives the design principle:

P.5.1.3: It is not possible to delegate same right over same object to multiple delegatees, however one can delegate different rights over same object to multiple delegatees.

5.1.1.4 Monotonicity

Monotonicity refers to whether the previous holder of the right can exercise it after the allocation. If *Alice* delegates the edit rights over a track to *Bob*, she cannot modify it anymore but only *Bob* can do any changes[67]. This condition is important for tracing the social errors to maintain trust of an actor. Socially, full accountability for a track increases effort, and likewise a 'free' community with delegated rights participates more. If *Bob* knows that *Alice* can remove his contributions without his consent, he would be reluctant to produce as much as he could do freely. Also, if he knows that *Alice* is also responsible for doing the same job[68], both may rely on each other and the task could suffer and progress will be reduced. However, If *Alice* feels that the administration of track is not proper and appropriate actions are not taken by the track chair, she can un-delegate and take the track back.

Each delegation gives two roles for every delegated right over object, that is, Delegator *(Dgr)* and the Delegatee *(Dge)*, It means that given *x* rights associated with an object, the Replace $_{Use}$ model can create up to *2x* roles, which is also convenient for assigning rights to various roles. For actor centric implementations, the rights associated with them are

$$Rep_u\ (Dgr) \rightarrow \begin{matrix} UR(Dgr, O, \phi) \\ MR(Dgr, O, All) \end{matrix} \ldots [eq.\ 5.5]$$

and

$$Rep_u\ (Dge) \rightarrow \begin{matrix} UR(Dge, O, All) \\ MR(Dge, O, \phi) \end{matrix} \ldots [eq.\ 5.6]$$

[66] Delegation in literature allows the user to delegate one role/permission to multiple users; however it is not supported by current online social interactions scenarios, so the proposed model has singular cardinality.

[67] The possibility when both can exercise the right is explored in Share $_{Use}$ (section 5.2).

[68] For example, sending reminder to reviewers.

where

$$Dgr_{UR} \cap Dge_{UR} = \phi \quad \ldots [eq.\ 5.7]$$

The above concludes that delegation is mutually exclusive, so if one right is delegated to the delegatee then it cannot be further exercised by the delegator. Only the delegator or the delegatee can exercise a right in a given state of the object at a particular time. This assumption is necessary to maintain the accountability of actions. It also provides full control over object and the delegatee will produce more as free control produces more. This gives the design principle:

P.5.1.4: The delegator cannot exercise the delegated right, without un-delegating it.

5.1.1.5 Depth

Depth deals with allowing the beneficiary to further pass on the allocated right. In the proposed model, if a delegatee gets no meta-rights, they cannot pass rights on, for example renting an apartment gives no right to sub-let[69]. Similarly, lending a book to another does not give them the right to on-lend it, for example if one loans a book to a person who loans it to another person who then loses it, who is accountable to the original owner? It is necessary to restrict the delegatee from passing on a delegated right[70] to maintain accountability and consistency in the system. In cases when delegatee is unable to perform the delegated task, he may give up the delegated right, which reverts back to the delegator – the owner, by the principle that all rights must be allocated.

In the above scenario, *Bob* cannot pass his track to *Carl* as *Alice* and *Carl* has no direct communication contract and *Carl* is not accountable to *Alice*. However, if for some reason, *Bob* is unable to continue his responsibilities as track chair, he can request *Alice* to take back the track and then she can delegate it to someone else. This gives the design principle:

P.5.1.5: Being delegated does not give the right to delegate.

[69] By this social logic, lessees can't sub-let, that is, delegate tenancy rights on to others.

[70] In traditional models, delegation is multi-step, which is against the ownership principles, so this model supports single-step delegation.

5.1.2 Replace $_{Use}$ process

The Replace $_{Use}$ model maps various components of the SAC model to each other and defines the delegation operation using the following set of rules:

a) The requestor VU_j belongs to the LR_i in the NS_i.

$$VU_j \rightarrow LR_i, \ \forall \ LR_i \in NS_i \ \ldots [eq.5.8]$$

b) The requested object O is classified into the same OC_τ.

$$O \rightarrow OC_\tau, \forall \ OC_\tau \in NS_i \ldots [eq.5.9]$$

c) The assignment of Dge role to VU_j and Dgr role to SH_i.

$$\begin{matrix} VU_j \rightarrow Dge \\ SH_i \rightarrow Dgr \end{matrix}, \forall \ Dgr, Dge \in LR_i \ldots [eq.5.10]$$

d) The addition of use-rights to Dge, and their removal from Dgr.

$$\begin{matrix} UR \rightarrow Dge \\ \neg UR \rightarrow Dgr \end{matrix}, \forall \, UR \rightarrow O \ldots [eq.5.11]$$

e) Meta-rights remain with the Dgr.

$$MR \rightarrow Dgr, \forall \, MR \rightarrow O \ldots [eq.5.12]$$

Given a $NS\,i$, a delegation access request is only granted if the virtual user VU_j acquires a local role LR_i in namespace NS_i. Also the object O is mapped in object class OC with the privacy label τ. Further, VU_j is assigned to the delegatee role Dge, which has the use rights over the object O.

5.1.3 Definition

Delegation is an operation that replaces the active actor set with another actor, that is, delegatee. It can be defined as follows:

NS, SH, VU, Opr, LR, Dge, Dgr, UR and O are sets of Namespaces, Stakeholders, Virtual Users, Operations, Local Role, Delegatee, Delegator, Use-Rights and Objects respectively.

- A one-to-many stakeholder to delegator role assignment relation is given by

$$SH\_Dgr \subseteq SH \times Dgr$$

- $assigned\_delegator: (sh:SH) \rightarrow 2^{Dgr}$ is a function derived from SH_Dgr, for mapping of any power set of delegator role onto stakeholder, which can be ϕ

(empty) or the SH for the delegation operation based on the current state of the object.

$$assigned\_delegator(sh) = \{sh \in SH, dgr \in Dgr | (sh, dgr) \in SH\_Dgr, \forall SH, Dgr \in NS_i\}$$

- A one-to-many virtual user to delegatee role assignment relation is given by

$$VU\_Dge \subseteq VU \times Dge$$

- $assigned\_delegatee: (vu: VU) \rightarrow 2^{Dge}$ is a function derived from VU_Dge, for mapping of delegatee role over objects onto a virtual user.

$$assigned\_delegatee\ (vu) = \{vu \in VU, dge \in Dge | (vu, dge) \in VU\_Dge, \forall\ VU, Dge \in NS_i\}$$

- $AC = 2^{(opr \times O)}$ is the set of attestation certificates for use-rights.
- A many-to-many delegatee role to attestation certificate assignment relation is given by

$$Dge\_AC \subseteq Dge \times AC$$

- $delegatee\_rights: (dge: LR) \rightarrow 2^{AC}$ is a function derived from Dge_AC, for mapping each Delegatee role to a set of attestation certificates.

$$delegatee\_rights\ (dge) = \{dge \in LR | (dge, ac) \in Dge\_AC, \forall\ LR \in NS_i\}$$

- A many-to-many delegator role to attestation certificates assignment relation is given by

$$Dgr\_AC \subseteq Dgr \times AC$$

- $delegator\_rights: (dgr: LR) \rightarrow 2^{AC}$ is a function derived from Dgr_AC, for mapping each Delegator role to a set of attestation certificates, which are not delegated to Dge.

$$delegator\_rights\ (dgr) = \{dgr \in LR | (dgr, AC) \in Dgr\_AC, \forall\ LR \in NS_i\}$$

- As all the rights over an object can be delegated to one or more delegatees, it gives the concept of active users over an object for a particular state. It can be calculated by equation

$$Active\ Users = Dgr \cup Dge$$

and the rights for these roles over the object is mutually exclusive, such that

$$Dgr\_UR \cap Dge\_UR = \phi$$

which shows that the rights of delegatee and delegator are disjoint for the same

object.

Access Grant

A virtual user *uv* can perform an operation *opr* over an object *O* under the delegation model only if there exist a mapping of vu_j for the delegatee role over the object, and the delegatee role is authorized for the requested right by having the desired *ac*, so making vu_j a member of active user set, under the following relation:

$$vu_j{:}VU, AU{:}Active\ Users, ac{:}\ AC\ \forall\ VU, AU \in NS_i$$

$$vu_j \in assigned\_delegatee\ (vu_j) \wedge ac \in delegatee\_rights(dge) \Longrightarrow vu_j \in AU$$

5.1.4 Rights analysis

This section describes the modification of rights for various roles of parent, offspring and general public associated with the delegated object. Consider the conference and track example, the delegated track was not moved from one conference to another, but remained in the same space – conference space, so the rights of objects' parent, offspring and general public remain the same. The rights of the delegator, that is, the conference chair, are reduced from the complete set of rights to a reduced set after giving it to delegatee, that is, the track chair. The delegatee rights are increased from none to the delegated set.

Below is table 5.1 for the roles of delegator and delegatee, and their rights over parent, delegated, and child objects.

	Delegator			**Delegatee**		
	Parent	**Delegated**	**Child**	**Parent**	**Delegated**	**Child**
Before Rep Use	UR(V)	UR(V,D,E) MR(V,D,E)	UR(V, D)	D/C	D/C	D/C
After Rep Use	UR(V)	UR(V,D) MR(V,D,E)	UR(V)	UR(V)	UR(V,D,E)	UR(V,D)
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively D/C represents don't care and depends on Delegatee's previous role in the space						

Table 5.1: Delegator and delegatee rights over different objects before and after delegation

Below is table 5.2 illustrating different rights for different roles associated with the object before and after delegation.

	Parent	Delegator	Delegatee	Offspring	G. Public
Before Rep $_{Use}$	UR(V, D)	UR(V,D,E) MR(V,D,E)	D/C	UR(V)	D/C
After Rep $_{Use}$	UR(V, D)	UR(V,D) MR(V,D,E)	UR(V,D,E)	UR(V)	D/C
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively D/C represents don't care and depends on Delegatee's previous role in the space, or space configuration for G. Public role					

Table 5.2: Rights for different roles associated with the object before and after delegation

A complete analysis of rights for different roles based on table 5.1 and 5.2 for the Replace $_{Use}$ model is as follows:

a) After Replace $_{Use}$, the delegator can only exercise the view right over the object for accountability reasons. The meta-right to revoke the delegation also remains with the delegator.
b) The view right of delegator over the parent object spaces remains the same, to enter the space.
c) The rights of delegator over child object also reduce from view and delete to view only.
d) The delegatee gets the delegated use-rights over the object, but does not get the meta-rights.
e) The delegatee gets the view right over the parent space.
f) The delegatee also gets the view and delete rights over the child objects.
g) The model does not change the space of the object, so parent's rights over the object remain the same, that is, they can view and delete the object.
h) Offspring role is associated with object, so its rights after the delegation of parent object are not changed and they can view/enter the parent space.
i) General public role remains the same for the object and so its rights, which depends on the parent and object configurations.

5.1.5 Design principles

The above discussion gives the following design principles for the Replace $_{Use}$ model.

a) The model addresses domain based delegations, where rights are associated with objects.
b) The delegator can only delegate the rights over the object for which they have the meta-rights.
c) Delegation gives use-rights, but meta-rights remain with the delegator.
d) The consent of delegator as well as of delegatee is required for the delegation.
e) A delegator can delegate some/all of his use-rights to the delegatee.
f) A delegator can delegate different rights over an object to multiple delegatees, but cannot delegate same right over the same object to multiple delegatees.
g) The delegator cannot exercise the delegated right.
h) The delegatee cannot further delegate the delegated rights.
i) The rights of delegator and delegatee are changed based on the delegated rights, but the rights of parent, offspring and general public role over the object remain the same.

5.1.6 Revocation

Revocation of rights is as important as it allocation in real life because granting rights is not static but dynamic in nature (Barka, 2002; Crampton & Khambhammettu, 2006, 2008; Ruan & Varadharajan, 2010). Friends in roles, employees in organizations and virtual users in events are changed with respect to time and their responsibilities (Barka, 2002; Barka & Sandhu, 2000a). In the running scenario, the conference may run for a certain time period and even during that time period it may have various phases, like submission phase, review phase, publishing phase, printing phase and the conference running phase. So the role of track chair for the reviewing phase can be assigned to *Bob*, but during the printing phase, *Carl* or *David* may replace him for the same track. This section will cover the possible ways in which a delegated right can be revoked from the delegatee.

A right allocation is revocable if the initiating party keeps the meta-rights and delegation allocates use-rights to the delegatee, but the meta-rights remain with the delegator, so it is revocable, for example a system administrator who delegates rights can take back the top

system priority (Gaaloul et al., 2008). In this case, the delegatee is responsible for the object, but the delegator is responsible for the delegatee, and can revoke their permission at any time. As the delegator keeps the meta-rights, they have the right to reallocate the delegation to anybody else or take it back, which gives the principle:

P.5.1.6: Delegator can revoke the delegation.

There are three types of revocation supported in the Replace $_{Use}$ model.

5.1.6.1 Self revocation

The delegator revokes the delegation based on inappropriate use or at will. Similarly if the delegatee cannot perform the task, they can withdraw from the delegated rights, in which case all the delegated rights previously delegated to that particular delegatee are returned to the delegator. In the above scenario, *Alice* can revoke the track chair position from *Bob* at any time at will.

Allowing self-revocation gives the freedom to the delegatee to work under one delegator and to fulfill the expectations of only one person. It also gives the sense of authority to the delegator that the object still belongs to them even after delegating it, and it can be taken back at any time if the requirements are not fulfilled by the delegatee.

5.1.6.2 Time based revocation

At the time of delegating the right, the delegator can assign a time-stamp with the delegated right for the proposed lifespan of the delegation. When the time-stamp expires, the delegation is automatically revoked from the delegatee and the delegator retrieves all the delegated rights over the object. Time based revocation provides a tool to the delegator so he does not need to worry about revoking each delegation. In the above scenario, *Alice* can set a time-stamp with the delegation so it would be revoked at the end of the review phase.

Time based revocation provides the facility to revoke the delegation without the involvement of the delegator. It gives the delegator the ease of not having to remember to revoke each delegation over his objects. In addition, it gives the delegatee incentive to properly manage the object and the assigned tasks within the time frame.

5.1.6.3 Rule based revocation

At the time of consent from the delegator and the delegatee, a set of rules are defined for the use of object, which is known as the delegation contract. If the delegatee violates any of the conditions, the delegation is revoked automatically and the delegator gets back all the rights. In the above scenario, *Alice* can define the rule that *Bob* can only review the papers but cannot delete any paper from the system, so as soon as *Bob* tries to delete any paper, the track chair delegation is revoked automatically.

Rule based revocation ensures the proper usage of the delegated rights and the delegator need not worry about tracking the delegated object. It gives the delegatee a boundary to work within as they know about their limits. This option tracks and maintains the consistency of the delegated object.

5.1.6.4 The Replace $_{Use}$ revoke process

The revocation removes the delegated use-rights *UR* from the delegatee *Dge* and adds them back to the delegator *Dgr*.

$$\begin{matrix} UR \rightarrow Dgr_i \\ \neg UR \rightarrow Dge_i \end{matrix} \quad \forall\, UR \rightarrow O \ldots [eq.\ 5.13]$$

which reverts all the delegated rights previously delegated to the particular delegatee back to the delegator. It also takes back the *Dge* role from VU_j and *Dgr* role from SH_i.

$$\begin{matrix} Dgr \rightarrow SH_i \\ \neg Dge \rightarrow VU_j \end{matrix} \quad \forall\ Dgr, Dge \in LR \wedge SH_i, VU_j \in NS_i \ \ldots [eq.\ 5.14]$$

5.1.7 Summary of Replace $_{Use}$

The difference between the characteristics of the proposed delegation model and the traditional delegation models can be seen in table 5.3.

In traditional role based delegation, the rights are associated with roles which are delegated to users, while the presented model associates rights with objects and the owner has the option to delegate subset of rights over subset of objects. Also, the Replace $_{Use}$ model is based on discretionary grounds, whereas the delegation model in RBAC was based on mandatory

grounds and a security officer is responsible for assigning roles (Sandhu & Munawer, 1998a, 1998b, 1999). This central administration approach cannot work in STS which are based on ownership and has the potential of millions of users. Furthermore, the delegation in RBAC allow the other role members to revoke the delegation (Barka & Sandhu, 2002), while the Replace $_{Use}$ model presented here only allows the owner to delegate/revoke the delegation.

Characteristic	Proposed Model	Traditional Models
Consent	Joint	Several
Totality	Partial	Total
Cardinality	Singular	Multiple
Monotonicity	Mutually Exdusive	Mutually Indusive
Depth	Single pass	Single pass
Revocation	Self/Time/Rule based	Member/Time based
Who can delegate	If one has the meta-right	If one has the use-right[71]
Who can revoke	Owner	Any member from the role

Table 5.3: Difference between proposed and traditional delegation models

To summarize, the Replace $_{Use}$ model allows the owner of an object to give the use-rights to some other actor to work on his/her behalf. The characteristics are given in section 5.1.1, which distinguish the model from other models of use-rights allocation. Section 5.1.2 explains the allocation process, followed by the logical definition of the model in section 5.1.3. Section 5.1.4 illustrates the rights analysis, the design principles are mentioned in section 5.1.5 while the revocation is discussed in section 5.1.6.

5.2 Share $_{Use}$ model

Share $_{Use}$ model allows the owner to allocate the use-rights to other actors while the rights of existing actors do not change (Y. Liu et al., 2009; Z. Zhang et al., 2008). In the use-rights triplet, it is done by adding an actor to the actors set, while the meta-rights triplet remains the same. The need for Share $_{Use}$ model arises when the owner of an object wants to give some rights over his object to others as well as keeps them, which in effect copies the rights. Entity operations like view are usually shared rather than replaced. It is used in scenarios when one actor wants to give away rights to others without giving them any responsibility, or giving the

[71] Member of the role.

responsibility to multiple actors simultaneously[72]. In sharing, the entire right is given completely, so any party can act alone as if they own it exclusively. Sharing a use-right adds an actor for use-rights but meta-rights remains with the original owner, so it can be revoked at any time. In the physical world, a couple's bank account where either can withdraw all the money is an example of sharing use-rights. In this example, the owner has the meta-right to close the account or to remove the beneficiary, but both can withdraw any amount of money as they own the 'withdraw' right completely.

Let's consider the scenario of a *'video on demand (VOD)'* website, which is a platform of sharing videos among the community on the basis of payment. The website allows the viewer to stream the specific video on the server at any time – provided that they are in the purchased window, and gives them personalized control[73] such as pause, forward and rewind. The video owner or the distributor adds videos and gives their view rights to the users who pay to watch that video. This view right cannot be delegated to a user as the same video can be seen by many users at the same time. Every video is a separate space with different settings, comments, rating and has various roles of distributor, commenters, likers, viewers and general public. The system contains thousands of videos viewed on a daily basis by millions of users around the world, so the rights need to be shared among all the viewers. The Replace $_{Use}$ model cannot be used as giving rights to single user will reduce the system's growth and business opportunities.

To map this scenario on the core SAC model, '*VOD'* system is the parent space for all the videos, owned by the system administrator. So, all the videos are at the same hierarchical level in the inheritance tree of the system, so parent and general public roles are same for every video. Videos are spaces with distributor as stakeholder and likes and comments are child objects of the video space. The parent[74] has the right to ban a video which violates the copyright or decency policy. The owner (distributor) has the right to decide its view access to the general public and/or to a specific set of viewers.

Now suppose that an actor *Alice* creates a new video in the system for distribution purposes, and wants to give *Bob* – an actor in viewer role, the view rights over the video. She wants *Bob*

[72] For example, to delegate a right to multiple actors at the same time.

[73] Against pay-per-view and video broadcast.

[74] The system administrator.

to be able to view the video for the time he has paid for, and he may comment or like it. She also wants to maintain her authority over the video and do not want him to delete the video, view it for an unspecified amount of time, or pass the view right to another user. Moreover, she wants to give this right for a specific number of views or time duration, and can take it back for next view if required. She then shares the view rights over her video with *Bob*, to include him in the allowed viewer for the video. This scenario is depicted in figure 5.2.

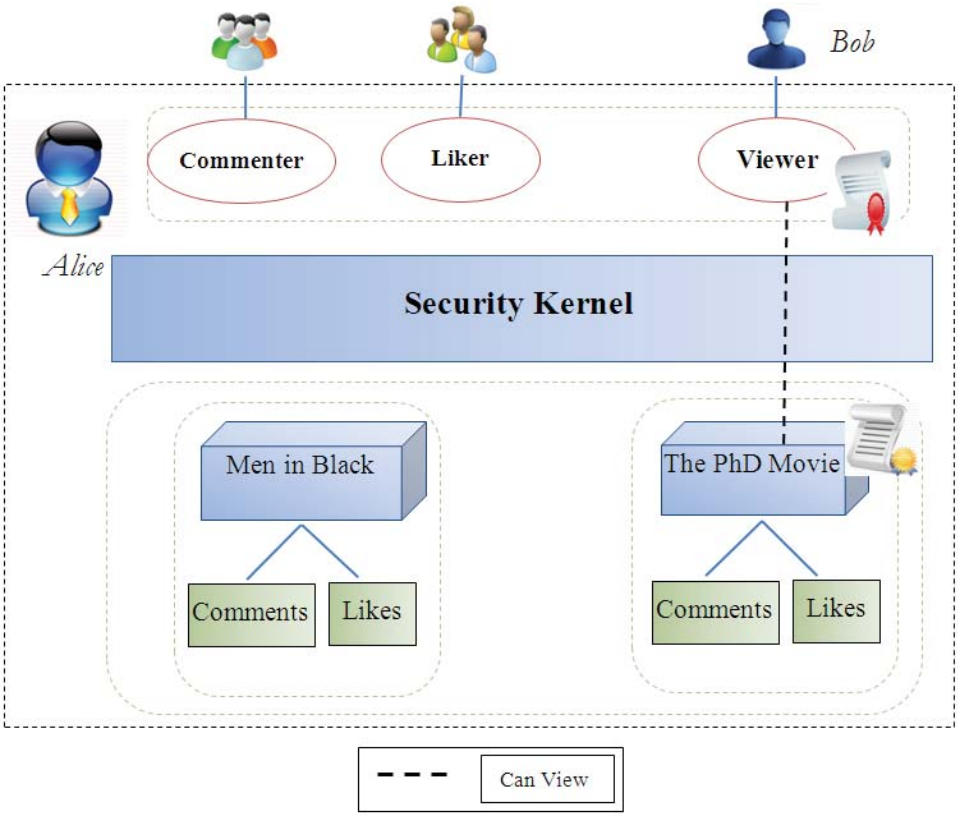

Figure 5.2: Sharing use-rights scenario depicting a VOD system

5.2.1 Characteristics of Share $_{Use}$

Following are some of the characteristics of the Share $_{Use}$ model that distinguish it from other models of use-rights. Some of these characteristics are explored in the literature (Barka & Sandhu, 2000b; L. Zhang et al., 2003; X. Zhang et al., 2003) for role to role delegation in RBAC (Ferraiolo & Kuhn, 1992; Ferraiolo et al., 2001; Sandhu et al., 1996). However, the values of these parameters are different in this research from the values proposed in the literature due to the nature of online social interactions and the ownership domain.

There are two basic axioms for this model, which are also kept consistent throughout all the other use-rights models in this research. First, the Share $_{Use}$ model presented in this research is domain based as rights are associated with objects. The second is that no one can share a use-right associated with an object which they do not own. So, it is not possible for *Alice* to assign *Bob* rights over a video, where she does not have the meta-right.

5.2.1.1 Consent

Allocating the right to view a video to *Bob* requires *Alice* to initiate the process as the system should enhance the trust of the owner that she has all the control over her object. *Alice* may want to add or delete a video at will and decide whether it is appropriate to display the video to a specific actor or not. However, *Bob* can request a video and pay if necessary, but it is not necessary to take the consent of *Bob* for every video he can watch. As the system may contain free videos available to the general public, this design option may have scalability issues, for example, it is not possible to ask the entire viewer and/or general public role for every free video, which can result in too many requests per video. Allowing someone to watch something is not a responsibility, so it does not require their consent.

This implies that the owner's consent is needed before any modification to the use-rights triplet and only the owner can decide whether to allow some actor over his object or not. It also implies that rights with no accountability to the beneficiary can be allocated without their consent, as others can use them if they wish, for example view and enter. So space owners can give entry and view rights without inconsistency. It gives the design principle:

P.5.2.1: Beneficiary's consent is not required for sharing of use-rights.

5.2.1.2 Totality

Totality deals with allocating the proper/improper subset of rights over proper/improper subset of authorized resources. The Share $_{Use}$ model allows *Alice* to decide how much use-rights she wants to give to *Bob* from her set of rights. She may want to allow him to only view the trailer or view the complete video, in which case the sharing is partial, or she may want to allow him to edit/delete the video as well, in which case the sharing is total[75]. This model

[75] As the illustrated scenario discusses a single video, it is implied that the model supports sharing partial set of resources.

operates on partial rights since total sharing reduces growth opportunities, as only all or none right can be assigned. Also, partial rights can be extended to the complete set, in cases where complete right set need to be shared by using

$$Shr_u(UR_{all}) = Shr_u(UR_1) \wedge Shr_u(UR_2) \wedge \ldots \wedge Shr_u(UR_n) \ldots [eq.5.15]$$

Generalizing the above implies that the Share $_{Use}$ model can operate on two design possibilities, that is, only complete sharing of use-rights where all the rights or none can be assigned to the beneficiary, or partial sharing where subset of rights held by the owner can be allocated. This model supports partial sharing as it is more beneficial in most cases and total allocation can be achieved using multiple partial sharing. This gives the design principle:

P.5.2.2: The owner can share proper/improper subset of their rights.

5.2.1.3 Cardinality

Cardinality deals with the number of beneficiaries in a particular rights allocation operation. In the above scenario, *Alice* can share the view right over her video to multiple actors say *Bob* and *Carl* at the same time. In this case the view right is given away to multiple actors who can simultaneously exercise the right, and it does not affect the object state, which gives

$$Shr_u \rightarrow (UR_i)\{(Bob \mid Carl) \mid Bob,Carl \in VU \, \forall \, VU \in NS_i\} \ldots [eq.5.16]$$

However if the edit right is given to multiple actors then the locking of objects, preservation of state and precedence of edits need to be recorded.

Generalizing the above implies that a right that does not change the object state can be given to multiple beneficiaries and it will not affect the object consistency. This feature is quite important for system growth and lack of it can result in decrease of financial benefits. However, in order to share edit rights (or rights associated with change) the system should support object locking and precedence of operation occurrences. The second case happens in collaborative software and Wikipedia, which shares edit rights among multiple actors. This gives the design principle:

P.5.2.3: It is possible to share same use-right with multiple beneficiaries.

5.2.1.4 Monotonicity

Monotonicity refers to whether the previous owner can exercise the right after the allocation. In the running example, if *Alice* shares view rights over her video with *Bob*, she can still view the video. The case where the owner cannot exercise the right is covered in Replace $_{Use}$ and thus this model covers the other cases where the owner and the beneficiary both can exercise the right[76]. If *Alice* has to decide every time she has a right, whether to keep it or to give it to *Bob*, the system would not grow to its full potential.

The Share $_{Use}$ model adds the beneficiary to the actor set for use-right, which gives

$$Shr_u \rightarrow UR(\{Alice \mid Bob\}, O, opr) \ldots [eq.5.17]$$

Sharing creates one role with the shared right associated with the object, that is, the Beneficiary *(Ben_u)*, where the rights of the owner do not change. It means that given x rights associated with an object, the Share $_{Use}$ model can create up to x roles, which is convenient for assigning rights to roles as well. For actor centric implementations, the rights associated with the owner and the beneficiary can be considered as

$$Shr_u(Ben_u) \rightarrow \begin{matrix} UR(\{Ben_u\}, O, opr) \\ MR(\{Ben_u\}, O, \phi) \end{matrix} \ldots [eq.5.18]$$

and

$$Shr_u(Owner) \rightarrow \begin{matrix} UR(\{Owner\}, O, opr) \\ MR(\{Owner\}, O, All) \end{matrix} \ldots [eq.5.19]$$

The above concludes that the sharing of use-rights is mutually inclusive, so if a right is shared between two actors, then both can exercise it as they have the complete right. This assumption allows many useful scenarios like YouTube video, Wikipedia articles, sharing of one research paper among various reviewers or in general sharing of any online object viewed by multiple actors. Another insight drawn is that the rights which do not affect the system state can be mutually inclusive while the rights modifying the object state needs to be mutually exclusive

[76] The other case when the owner and the beneficiary both have to agree to exercise a right is covered in Merge $_{Use}$ (Section 5.3).

(or the system should support some locking mechanism), if exercised by multiple actors simultaneously. This gives the design principle:

P.5.2.4: Both the owner and the beneficiary can exercise the shared use-right.

5.2.1.5 Depth

Depth deals with allowing the beneficiary to further pass on the allocated right. The Share $_{Use}$ model does not give any meta-right to the beneficiary, so they cannot pass the right to others, for example you cannot invite guests on someone else's party. This condition maintains the owner's authority over the object, where only they can decide who may have rights over it[77]. As the Share $_{Use}$ model does not account the beneficiary for any responsibility, this model is single step, so the beneficiary may exercise the right but cannot pass it on to another actor.

In the above scenario, *Bob* cannot pass the view right to *Carl* as he does not have the meta-right to do so. However, *Carl* can get the right by directly requesting it from *Alice*, who may allow or deny. As the Share $_{Use}$ model gives no meta-rights, only *Alice* can perform any rights assignment, which can be seen by

$$Shr_u \rightarrow \begin{matrix} UR(\{Owner \mid Ben_u\}, O, opr) \\ MR(Owner, O, All) \end{matrix} \dots [eq.5.20]$$

This gives the following design principle:

P.5.2.5: Sharing use-rights does not give the right to further share it.

5.2.2 Share $_{Use}$ process

The Share $_{Use}$ model maps various components of the SAC model to each other and defines the share operation using the following set of rules:

a) The VU_j belongs to the LR_i in the NS_i.

$$VU_j \rightarrow LR_i, \ \forall \ LR_i \in NS_i \dots [eq.5.21]$$

b) The requested object O is classified into the same OC_τ.

[77] Traditional rights allocation models do not maintain this accountability and so any user can pass the rights after acquiring it.

$$O \rightarrow OC_\tau, \forall\ OC_\tau \in NS_i \ldots [eq.5.22]$$

c) The assignment of Ben_u^i role to VU_j.

$$VU_j \rightarrow Ben_u^i, \forall\, Ben_u^i \in LR_i \ldots [eq.5.23]$$

d) The sharing of use-rights with the Ben_u^i.

$$Shr_u \rightarrow \begin{matrix} UR \rightarrow Owner_i \\ UR \rightarrow Ben_u^i \end{matrix} \forall\, UR \rightarrow O \ldots [eq.5.24]$$

e) Meta-rights remain with the $Owner_i$.

$$MR \rightarrow Owner_i, \forall\, MR \rightarrow O \ldots [eq.5.25]$$

Given a *NS i*, a share access request is only granted if the virtual user VU_j acquires a local role LR_i in namespace NS_i. Also the object *O* is mapped in object class *OC* with the privacy label τ. Further, VU_j is assigned the beneficiary role Ben, which has the use-rights over the object *O*.

5.2.3 Definition

Sharing is an operation that adds another actor to the actor set for a use-right. It can be defined as follows:

NS, VU, Opr, LR, Ben_U, *UR* and *O* are sets of Namespaces, Virtual Users, Operations, Local Roles, Beneficiary, Use-Rights and Objects respectively.

- A many-to-many virtual user to $Beneficiary_{U_u}$ role assignment relation is given by
$$VU\_Ben_U \subseteq VU \times Ben_u$$
- $assigned\_Ben_U: (ben_U: LR) \rightarrow 2^{VU}$ is a function derived from VU_Ben_U for mapping of Ben_U role over object to a set of virtual users in a namespace.
$$assigned\_Ben_u\ (ben_U) = \{vu \in VU, ben_u \in LR | (vu, ben_u) \in VU\_Ben_U, \forall\ VU, LR \in NS_i\}$$
- $AC = 2^{(opr \times O)}$ is the set of attestation certificates.
- A many-to-many Ben_U role to attestation certificate assignment relation is given by
$$Ben_U\_AC \subseteq Ben_u \times AC$$
- $Ben_U\_rights: (ben_U: LR) \rightarrow 2^{AC}$ is a function derived from Ben_U_AC for mapping each Beneficiary role to a set of attestation certificates.
$$Ben_U\_rights\ (ben_U) = \{ben_U \in LR | (ben_U, ac) \in Ben_U\_AC, \forall\ LR \in NS_i\}$$
- As all the rights over an object can be shared among one or more beneficiaries, it

gives the concept of active users over an object state and can be calculated as

$$Active\ Users = Owner \cup Ben_U$$

where Ben_x is the set of all beneficiaries for a right over an object.

Access Grant

Using the above functions, a virtual user vu_j can perform an operation opr over an object O, under the Share $_{Use}$ model only if there exist a mapping of vu_j for the Ben_x role over the object, and the Ben_x role is authorized for the requested ac, so making vu_j a member of active user set, under the following relation:

$$vu_j{:}VU, AU{:}Active\ Users, ac{:}AC\ \forall VU, AU \in NS_i$$

$$vu_j \in assigned\_Ben_u\ (vu_j) \wedge ac \in Ben_u\_rights(Ben_u) \Longrightarrow vu_j \in AU$$

5.2.4 Rights analysis

This section describes the modification of rights for various roles of parent, offspring and general public associated with the shared object. Consider the VOD system example: the viewed video is not moved from one space to another, but remains in the same space – system/genre, so the rights of its owner, parent, offspring and general public role remain the same. The beneficiary (viewer) rights are increased from none to the some subset, while the rights of owner (the distributor) are not affected by the sharing of use-rights.

Below is table 5.4 for the roles of owner and beneficiary, and their rights over parent, shared, and child objects.

	Owner			**Ben** $_{Use}$		
	Parent	**Shr. Obj**	**Child**	**Parent**	**Shr. Obj**	**Child**
Before Shr $_{Use}$	UR(V)	UR(V,D,E) MR(V,D,E)	UR(V,D)	D/C	D/C	D/C
After Shr $_{Use}$	UR(V)	UR(V,D,E) MR(V,D,E)	UR(V,D)	UR(V)	UR(V,D,E)	UR(V,D)

V, D and E represents view, delete and edit respectively

UR and MR represents use-rights and meta-rights respectively

Shr. Obj. represents the object with shared rights

Ben $_{Use}$ represents the beneficiary of use-rights allocation

D/C represents don't care and depends on beneficiary previous role in the space

Table 5.4: Owner and beneficiary rights over different objects before and after the Share $_{Use}$ model

Below is table 5.5 illustrating different rights of different roles associated with object before and after the Share $_{Use}$ model.

	Parent	Owner	Ben $_{Use}$	Offspring	G. Public
Before Shr $_{Use}$	UR(V, D)	UR(V,D,E) MR(V,D,E)	D/C	UR(V)	D/C
After Shr $_{Use}$	UR(V, D)	UR(V,D,E) MR(V,D,E)	UR(V,D,E)	UR(V)	D/C

V, D and E represents view, delete and edit respectively.
UR and MR represents use-rights and meta-rights respectively
Ben $_{Use}$ represents the beneficiary of use-rights allocation
D/C represents don't care and depends on beneficiary previous role in the space, or space configuration for G. Public role.

Table 5.5: Rights for different roles associated with object before and after the Share $_{Use}$ model

A complete analysis of rights for different roles based on table 5.4 and table 5.5 for the Share $_{Use}$ model is as follows:

a) After Share $_{Use}$, the owner can exercise all the rights over the object as he used to exercise before sharing. The meta-rights also remain with the owner, so he can also revoke the rights from the beneficiary at any time.
b) The rights of owner over parent object spaces remain the same, so they have view rights over them to enter.
c) The rights of owner over child object remain the same, that is, they can view and delete the child object.
d) The beneficiary gets the shared rights over the object except meta-rights.
e) The beneficiary gets the view right over the parent objects to enter the space.
f) The beneficiary gets the view and delete rights over the child object.
g) The Share $_{Use}$ model does not change the space of the object so parent's rights over object remain the same, that is, they can view and delete the object.
h) Offspring role is associated with object so its rights after the sharing of rights over parent object are not changed, that is, they can view/enter the parent object space.
i) General public role, its active users and their rights associated with the video remains the same, which depends on the system and the video space configurations.

5.2.5 Design principles

The above discussion gives the following design principles for the Share $_{Use}$ model.

a) The model addresses domain based sharing, where rights are associated with objects.
b) The owner can only share the rights over the object for which they have the meta-rights.
c) Sharing gives use-rights, but meta-rights remains with the owner.
d) The Share $_{Use}$ model only requires the owner's consent, but can operate without the consent of the beneficiary.
e) The owner can share some/all of his use-rights.
f) The owner can share the same right over the same object to multiple beneficiaries.
g) Both the owner and the beneficiary can exercise the right at the same time.
h) The beneficiary cannot pass on the right.
i) The rights of owner, parent, offspring and general public role remain the same after sharing the right with the beneficiary.

5.2.6 Revocation

Rights' sharing is a core component of STS as they are used to share millions of objects on a daily basis. However, sharing is not static but dynamic in nature due to removal of objects and frequent change in access permissions for roles (Barka, 2002; Crampton & Khambhammettu, 2006, 2008; Ruan & Varadharajan, 2010). In the running example, *Alice* – the distributor in *'VOD'* system, may assign view rights to *Bob* over videos he has purchased for a certain amount of time agreed upon by the purchased contract. So the rights over *'The PhD Movie'* can be assigned to *Bob* for, let's say, two days and after that it may get expired. This section covers the possible ways in which a shared right can be revoked from the beneficiary. As the owner keeps the meta-rights, *Alice* has the authority to take back the view right over her video from *Bob* (or the whole community) by revoking the right or by deleting the video at any time.

A right allocation is revocable if the initiating party keeps the meta-rights and the Share $_{Use}$ model allocates use-rights to the beneficiary, but the meta-rights remain with the owner, so it is revocable. This gives the following principle:

P.5.2.6: The owner can revoke the shared right.

There are three types of revocation supported in the Share $_{Use}$ model.

5.2.6.1 Self revocation

The owner revokes the right based on inappropriate use or at will. In the above scenario, *Alice* can revoke the view rights over her video from *Bob* at any time she wills. Self-revocation gives the sense of authority to the owner that the shared object still belongs to them as they can manage the rights over it, or take them back at any time.

5.2.6.2 Time based revocation

At the time of sharing the right, the owner can assign a time-stamp with the shared right for the proposed lifespan of the sharing. When the time-stamp expires, the sharing is automatically revoked from the beneficiary. Time based revocation provides a tool to the owner so he does not need to worry about revoking each right for each object. In the above scenario, *Alice* can set a time-stamp on the right over the video so it is revoked when the payment window expires. Time based revocation provides the facility to revoke the sharing without the involvement of the owner. It is especially beneficial in the environment of *'VOD'* systems, where payment is the measure of sharing rights for a time constraint window.

5.2.6.3 Rule based revocation

At the time of sharing, a set of rules is defined for the use of the object, which is known as the sharing contract. If the beneficiary violates any of the conditions, the sharing is revoked automatically. In the above scenario, *Alice* can define the rule that *Bob* can only view the video for three times and so as soon as *Bob* tries to watch it fourth time the view rights are automatically revoked.

Rule based revocation allows the owner to ensure the proper usage of the shared right so he does not need to track the object and rights for each beneficiary. This option allows worry-free tracking and maintains the consistency in the Share $_{Use}$ model.

5.2.6.4 The Share $_{Use}$ revoke process

The revocation removes the use-rights from the beneficiary Ben_u.

$$\neg UR_{(Use)} \rightarrow Ben_u^i, \forall UR \rightarrow O \ldots [eq.5.26]$$

It also takes back the Ben_u role from VU_j .

$$\neg\, Ben_u^i \rightarrow\ VU_j\ ,\ |\ Ben_u^i \in LR\ \wedge\ VU_j \in NS_i \ldots [eq.5.27]$$

5.2.7 Summary of Share $_{Use}$

The proposed model outlines the sharing of rights in online social interactions. To summarize, the Share $_{Use}$ model allow the owner of an object to give the use-rights to another actor and keep them at the same time. The characteristics are given in section 5.2.1, which distinguish the model from other models of use-rights allocation. Section 5.2.2 explains the allocation process, followed by the logical definition of the model in section 5.2.3. Section 5.2.4 illustrates the rights analysis, the design principles are mentioned in section 5.2.5 while the revocation is discussed in section 5.2.6.

5.3 Merge $_{Use}$ model

Merge $_{Use}$ model allows the owner to allocate use-rights over his object among an actor set in a way that they all must agree to exercise that right (Guo & Georganas, 2002; Ilic, Michahelles, & Fleisch, 2007). In the use-rights triplet, it is done by adding another actor to the existing set of actors and then merging the right of all of them, while the meta-rights triplet remains the same. The need of Merge $_{Use}$ model arises when the owner of an object allows other actors to exercise some right over his object with the strict condition of joint consent (Wu, 2003). Sensitive operations which involve critical changes in the object state are usually merged rather than shared, and it requires multiple actors to collaborate in order to complete the task. Merge $_{Use}$ amalgamates the entity use-rights but the meta-rights remain with the owner and so it can be revoked at any time. The owner of an entity gives rights to others and also keeps the authority to finalize it, which in effect divides the right. For example, Microsoft Word track changes allow the beneficiary to modify but the final accept/reject decision remains with the owner. In rights merge, any participant can stop an act, but performing it requires the consent of the whole actor set. In the physical world, a bank loan process where the bank clerk and the manager both agree to give the loan to a specific customer is an example of merging use-rights. In this example, both the bank clerk and bank manager are responsible for the loan approval and one cannot complete the process without the consent of the other.

Let's consider the scenario of *collaborative software*[78], which are platforms designed to help team collaborations. This software is used for working in co-ordinated environments to achieve a common goal. For example, in an editing exercise, *'collaborative software'* provide functionality by coordinating all changes made on a document edited by multiple authors. The edit rights of all the authors are merged so one may modify the work and others may have a chance to look at the modifications and accept them if they agree. In case of disagreement, the change can be rejected or modified again and send back to the first author. The system cannot use the Share $_{Use}$ model as it would allow the second author to modify whatever they want without the consent of the first author, it cannot use the Replace $_{Use}$ model as the first author then cannot modify the same document, so a model where all the authors have the authority must be used to enhance community usage and productivity. Obviously, this kind of allocation is only applicable within trusted users and the owner takes all the responsibility if the merging goes wrong.

To map this scenario on the core SAC model, the *'collaborative software'* is the parent space of all the group activities going on including puzzle matching, documents including/excluding, local events and so on. All the documents are at the same hierarchical level in the inheritance tree of the system so parent and general public roles are the same for every document. Documents are spaces with owner as stakeholder and 'like' and 'comments' are child objects of the document space. The parent has the right to view and delete a document if necessary, while the owner has the right to decide its access to the general public, to viewer and/or to the group members.

Now suppose that an actor *Alice* creates a new document in the system and wants to give *Bob* – another actor, the edit rights over the document. She wants *Bob* to be able to edit the document but the editing should not be done without her consent. She also wants to maintain her authority so *Bob* cannot delete the file, cannot give access to others, and if needed she can revoke his access to the document. She then merges the editing rights over her document with *Bob* to allow him conditional edits. For such cases, merging the editing rights of two actors requires both of them to agree on any edits, as MS Word processing track-change functions currently try to do. This scenario is depicted in figure 5.3.

[78] Also known as groupware.

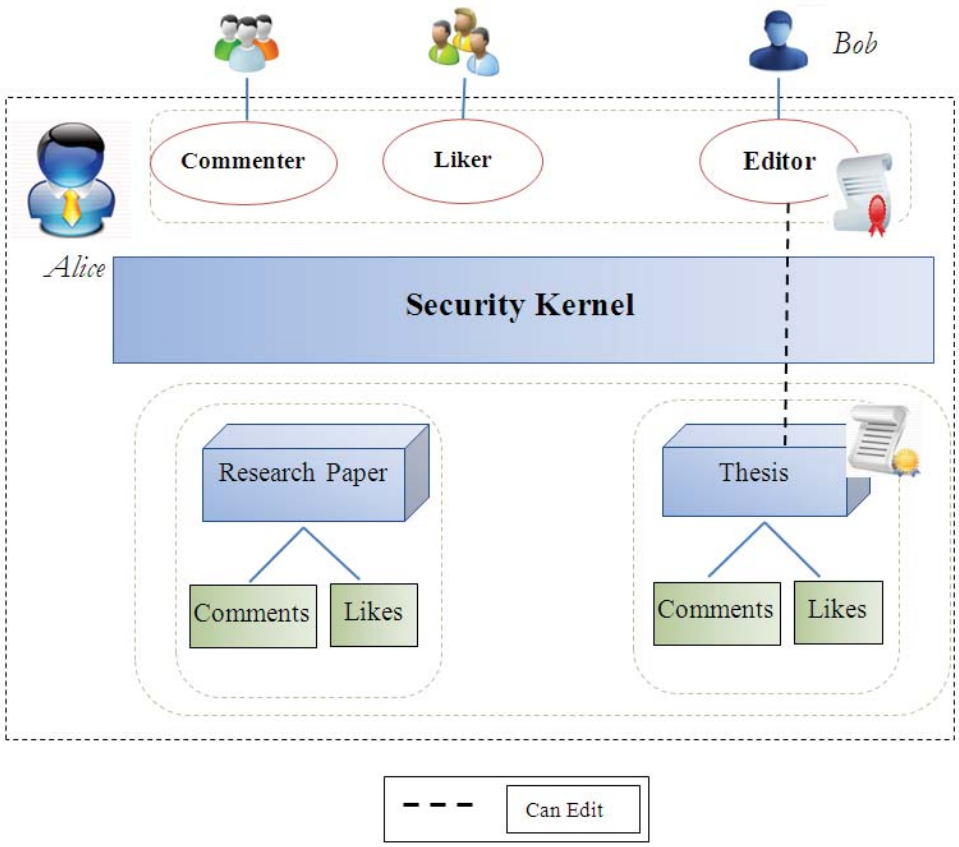

Figure 5.3: Merge use-rights scenario depicting collaborative software

5.3.1 Characteristics of Merge $_{Use}$

Following are some of the characteristics of the Merge $_{Use}$ model that distinguish it from other models of use-rights. Some of these characteristics are explored in the literature (Barka & Sandhu, 2000b; L. Zhang et al., 2003; X. Zhang et al., 2003) for role to role delegation in RBAC (Ferraiolo & Kuhn, 1992; Ferraiolo et al., 2001; Sandhu et al., 1996). However, the values of these parameters are different in this research from the values proposed in the literature due to the nature of online social interactions and the ownership domain.

There are two basic axioms for this model: First the Merge $_{Use}$ model presented in this research is domain based as rights are associated with objects. Second, no one can merge a use-right associated with an object that they do not own. So it is not possible for *Alice* to assign *Bob* the rights over another document, if she does not have the meta-right.

5.3.1.1 Consent

Allocating the edit right over a document to *Bob* requires *Alice* to initiate the process as the system should maintain the owner's authority and full control over her object. *Alice* may want to create or delete documents at will and decide whether it is appropriate to give the document access to a specific actor or not. Also, when the model merges the rights of *Alice* with *Bob,* his consent is required to execute the operation over the object, and it is necessary to ask *Bob* whether he wants to take the right or not. This would increase *Alice's* trust in the system and *Bob* will also be aware of his responsibility. If a right can be merged without the consents of the parties involved, the system will collapse due to unwilling authorities involved in performing a task, and the objects remain with no possible action, for example, assigning reviewers to papers without their consent.

The above implies that merging requires the consent of both parties and both are responsible for the right over the object. It gives the design principle:

P.5.3.1: Merging use-rights require the consent of the owner and the beneficiary.

This principle along with design principle 5.1.1 and 5.2.1 also gives the insight that rights which imply no real responsibility can be assigned freely without the consent of the beneficiary, but the rights imposing some responsibility require the consent of the beneficiary.

5.3.1.2 Totality

Totality deals with allocating the proper/improper subset of rights over proper/improper subset of authorized resources. In the above example, if *Alice* is the owner of the document, she decides how much rights she wants to assign to *Bob* from her set of rights. She may only want to merge the editing rights over the document with *Bob*, in which case the merging is of partial set of use-rights, or she may want to merge all the rights over the document with him, in which case the merge is total and applied on the complete set of use-rights[79]. Partial merging has some useful scenarios like only merging the accept/reject right among the reviewers of a conference, which can lose its effectiveness if only total is allowed as then the reviewers can modify the paper, or even delete it from the system.

[79] As the illustrated scenario discusses a single document, it is implied that the model supports merging of partial set of resources.

As the Merge $_{Use}$ model supports different use-right sets, the partial merge can be used for any subset of rights using

$$Mrg_u(UR_{all}) = Mrg_u(UR_1) \wedge Mrg_u(UR_2) \wedge \ldots \wedge Mrg_u(UR_n) \ldots [eq.5.28]$$

The above concludes that there are two design possibilities when merging the use-rights, that is, only complete use-rights set merging is supported so all the rights or none can be assigned to the beneficiary, or partial merge is supported so part of use-rights set held by the owner can be merged with the beneficiary. The Merge $_{Use}$ model supports partial merging as it has more useful scenarios and also total merging can be achieved using partials if needed, but the reverse is not possible. This gives the following design principle:

P.5.3.2: The owner can merge proper/improper subset of their rights.

5.3.1.3 Cardinality

Cardinality deals with the number of beneficiaries in a particular rights allocation operation. In the running example, *Alice* can merge the edit rights over her documents with multiple actors, say *Bob* and *Carl* at the same time. In this case, the same set of use-rights is merged with *Bob* and *Carl* and all need to coordinate with each other in order to perform the operation, which gives

$$Mrg_u \rightarrow (UR_i)\{(Alice\&Bob\&Carl) | Bob, Carl \in VU \wedge Alice \in SH, \forall\, VU, SH \in NS_i\} \ldots [eq.5.29]$$

Merging use-rights can raise the case when only one actor may veto to carry out the operation and thus cause the whole process to terminate. Generalizing the above statement implies that the same set of use-rights can be merged with multiple beneficiaries without concerns about the preservation of state. This feature is important in cases where maintaining the locks over objects may reduce the benefit of the system use. However, as the number of actors holding the merged right is increased, it becomes more difficult to modify the object as the agreement of the complete actor set needs to be considered for every single modification. This gives the design principle:

P.5.3.3: It is possible to merge the same use-right with multiple beneficiaries.

5.3.1.4 Monotonicity

Monotonicity refers to whether the previous owner can exercise the right after the allocation. In the above example, if *Alice* merges the edit rights over her document with *Bob*, she cannot modify it without his confirmation. The case where the initiating party cannot exercise the right at all is covered in the Replace $_{Use}$ model and the case when both the owner and the beneficiary can exercise the right at the same time is covered in the Share $_{Use}$ model, so this Merge $_{Use}$ model covers the options when the rights can be used by both actors but only with the joint consent. It adds the beneficiary to the actor set for a particular use-right and merges the right of the whole set, which gives

$$Mrg_u \rightarrow UR(\{Alice\,\&\,Bob\}, O, opr) \ldots [eq.5.30]$$

The Merge $_{Use}$ model creates two roles with the merged right associated with the object, that is, the Primary Owner *(PO)* and the Joint Beneficiary *($JBen_u$)*. It means that given *x* rights associated with an object, the Merge $_{Use}$ model can create up to *2x* roles, which is also convenient for assigning rights to various roles. For actor centric implementations, the rights associated with them can be considered as

$$Mrg_u(JBen_u) \rightarrow \begin{matrix} UR(^1/_N\{JBen_u\}, O, opr) \\ MR(JBen_u, O, \phi) \end{matrix} \ldots [eq.5.31]$$

and

$$Mrg_u(PO) \rightarrow \begin{matrix} UR(^1/_N\{PO\}, O, opr) \\ MR(PO, O, All) \end{matrix} \ldots [eq.5.32]$$

where N is the total number of joint beneficiaries for a given use right. It can be generalized that if rights of two actors are merged, then both of them have to co-ordinate to exercise the right. This assumption allows many useful scenarios like MS office, collaborative software, video conferencing, and so on, and gives the opportunity for collaboration even without the presence of object locking and rights precedence. This gives the design principle:

P.5.3.4: Joint consent of primary owner and joint beneficiary is required to exercise the merged use-right.

5.3.1.5 Depth

Depth deals with allowing the beneficiary to further pass on the allocated right. In the proposed model, as merge only modifies the use-rights triplet but the meta-rights remain with the owner, the beneficiary cannot pass the merged use-right to others. This condition maintains the owner's authority over the object since they retain the authority to decide who else can have the rights over the object or when to retrieve the merged right. The Merge $_{Use}$ model proposed in this research is single step, therefore the beneficiary cannot pass on the use-right to another actor.

In the above scenario, *Bob* cannot pass the edit rights over the document to *Carl* as he does not have the meta-right to do so. Use-rights merge leaves the meta-rights with *Alice* who can modify the use-rights triplet. This condition is necessary for user trust and gives the following design principle:

P.5.3.5: Merging use-rights does not allow the beneficiary to further allocate it.

5.3.2 Merge $_{Use}$ process

The Merge $_{Use}$ model maps various components of the SAC model to each other and defines the merge operation using the following set of rules:

a) The VU_j belongs to the LR_i in the NS_i.

$$VU_j \rightarrow LR_i, \ \forall \ LR_i \in NS_i \ldots [eq.\ 5.33]$$

b) The requested object O is classified into the same OC_τ.

$$O \ \rightarrow \ OC_\tau, \forall \ OC_\tau \in NS_i \ldots [eq.\ 5.34]$$

c) The assignment of $JBen_u^i$ role to VU_j and PO_i to SH_i.

$$\begin{matrix} VU_j \rightarrow JBen_i \\ SH_i \rightarrow PO_i \end{matrix} \forall \ PO_i, JBen_i \in LR_i \ldots [eq.\ 5.35]$$

d) The addition of use-rights to the $JBen_u^i$ and restricting PO_i and $JBen_u^i$ to act jointly.

$$Mrg_u \rightarrow \begin{matrix} 1/2\ UR \rightarrow PO_i \\ 1/2\ UR \rightarrow JBen_u^i \end{matrix} \forall\, UR \rightarrow O \ldots [eq.\ 5.36]$$

e) Meta-rights remain with the PO_i.

$$MR \rightarrow \ PO_i, \forall\, MR \rightarrow O \ldots [eq.\ 5.37]$$

Given a *NS i*, a merge access request is only granted if the virtual user VU_j acquires a local role LR_i in namespace NS_i. Also the object *O* is mapped in object class *OC* with the privacy label τ. Further, VU_j is assigned to the joint beneficiary role $JBen_n$ which has the joint use rights over the object *O*.

5.3.3 Definition

Merge is an operation that adds another actor to the actor set for a use-right and merges the use-right of the entire actor set. It can be defined as follows:

NS, SH, VU, Opr, LR, JBen$_u$, *PO, UR* and *O* are sets of Namespaces, Stakeholders, Virtual Users, Operations, Local Roles, Joint Beneficiary, Primary Owner, Use-Rights and Objects respectively.

- A one-to-many stakeholder to Primary Owner role assignment relation is given by

$$SH\_PO \subseteq SH \times PO$$

- $assigned\_PO: (sh: SH) \rightarrow 2^{PO}$ is a function derived from *SH_PO*, for mapping of primary owner role onto stakeholder, which can be ϕ (empty) or the *SH* for the merge model based on the current state of the object.

$$assigned\_PO(sh) = \{sh \in SH, po \in PO | (sh, po) \in SH\_PO, \forall LR \in NS_i\}$$

- A many-to-many virtual user to Joint Beneficiary $_{Use}$ role assignment relation is given by

$$VU\_JBen_U \subseteq VU \times JBen_U$$

- $assigned\_JBen_U: (Jben_U: LR) \rightarrow 2^{VU}$ is a function derived from VU_JBen_U, for mapping of $JBen_U$ role over object to a set of virtual users.

$$assigned\_JBen_U\ (Jben_U) = \{vu \in VU, Jben_U \in LR | (vu, Jben_U) \in VU\_JBen_U, \forall\ VU, LR \in NS_i\}$$

- $AC = 2^{(opr \times O)}$ is the set of attestation certificates.
- A many-to-many *PO* role to attestation certificate assignment relation is given by

$$PO\_AC \subseteq PO \times AC$$

- $PO\_rights: (po: LR) \rightarrow 2^{AC}$ is a function derived from *PO_AC*, for mapping each primary owner role to a set of attestation certificates.

$$PO\_rights\ (po) = \{po \in LR | (po, ac) \in PO\_AC, \forall\ LR \in NS_i\}$$

- A many-to-many $JBen_U$ role to attestation certificate assignment relation is given by

$$JBen_u\_AC \subseteq JBen_u \times AC$$

- $Ben_U\_rights: (Jben_U{:}LR) \rightarrow 2^{AC}$ is a function derived from $JBen_U\_AC$, for mapping each Joint Beneficiary role to a set of attestation certificates.

$$JBen_U\_rights\ (Jben_U) = \{Jben_U \in LR | (Jben_U, AC) \in JBen_U\_AC, \forall\ LR \in NS_i\}$$

- As all the rights over an object can be merged with one or more beneficiaries, it gives the concept of active users over an object state, and can be calculated by

$$Active\ Users = PO\ \cup JBen_U$$

Access Grant

Using the above functions, a virtual user *vu* can perform an operation *opr* over an object *O*, under the Merge $_{Use}$ model only if there exist a mapping of vu_j for the $JBen_x$ role over the object, and the $JBen_x$ role is authorized for the requested *ac*, so making vu_j a member of active user set, under the following relation:

$$vu_j{:}VU, AU{:}Active\ Users, ac{:}\ AC\ \forall VU, AU \in NS_i$$

$$vu_j \in\ assigned\_JBen_U\ (Jben_U) \wedge ac \in JBen_U\_rights(JBen_U) \Longrightarrow vu_j \in AU$$

but an operation can only be performed with the consent of *PO*.

5.3.4 Rights analysis

This section describes the modification of rights for various roles of parent, offspring and general public associated with the object. Consider the collaborative software system example, the document is not moved from one system to another but remain in the same space – the document category of the system, so the rights of its owner, parent, offspring and general public role remain the same. The joint beneficiary – *Bob's* rights are increased from none to a subset and the rights of primary owner – *Alice*, are reduced as now she needs to act together with the joint beneficiary.

Below is table 5.6 for the roles of primary owner and joint beneficiary, and their rights over parent, merged[80], and child objects.

[80] The focused object in the context of rights allocation.

	PO			JBen $_{Use}$		
	Parent	**Mrg. Obj**	**Child**	**Parent**	**Mrg. Obj**	**Child**
Before Mrg $_{Use}$	UR(V)	UR(V,D,E) MR(V,D,E)	UR(V,D)	D/C	D/C	D/C
After Mrg $_{Use}$	UR(V)	½UR(V,D,E) MR(V,D,E)	½UR(VD)	UR(V)	½UR(V,D,E)	½UR(V,D)

V, D and E represents view, delete and edit respectively
UR and MR represents use-rights and meta-rights respectively
PO represents primary owner, while JBen $_{Use}$ represents the joint beneficiary of use-rights
Mrg. Obj. represents the object with merged rights
D/C represents don't care and depends on beneficiary previous role in the space

Table 5.6: Owner and beneficiary rights over different objects before and after the Merge $_{Use}$ model

Below is table 5.7 illustrating different rights of different roles associated with object before and after the Merge $_{Use}$ model.

	Parent	**PO**	**JBen $_{Use}$**	**Offspring**	**G. Public**
Before Mrg $_{Use}$	UR(V,D)	UR(V,D,E), MR(V,D,E)	D/C	UR(V)	D/C
After Mrg $_{Use}$	UR(V,D)	½ UR(V,D,E), MR(V,D,E)	½ UR(V,D,E)	UR(V)	D/C

V, D and E represents view, delete and edit respectively
UR and MR represents use-rights and meta-rights respectively
PO represents primary owner, while JBen $_{Use}$ represents the joint beneficiary of use-rights
D/C represents don't care and depends on beneficiary previous role in the space, or space configuration for G. Public role.

Table 5.7: Rights for different roles associated with object before and after the Merge $_{Use}$ model

A complete analysis of rights for different roles based on table 5.6 and 5.7 for the Merge $_{Use}$ model is as follows:

a) After Merge $_{Use}$, the primary owner rights over the object are reduced from having all rights as single authority to joint authority. The meta-rights remain with the owner so they can revoke the right at any time.
b) The rights of the primary owner over parent object spaces remain the same, that is, they can enter the space as before.

c) The primary owner's rights over child objects are also merged with the joint beneficiary, so they cannot delete anything alone.
d) The rights of joint beneficiary over the object are increased as they can exercise some of the rights with the consent of the primary owner except meta-rights.
e) The joint beneficiary gets the view right over the parent spaces to enter the space.
f) The joint beneficiary also gets the merged delete right over the child objects.
g) Merging of use-rights does not change the object space, so parent's rights over the object remain the same, that is, they can view and delete the object.
h) Offspring role is associated with object, so its rights after the merging of parent object rights are not changed and they can still view/enter the parent space.
i) General public role, its active users and their rights associated with document remains the same, which depends on the parent and object configurations.

5.3.5 Design principles

The above discussion gives the following design principles for the Merge $_{Use}$ model.

a) The model addresses domain based use-rights merge where rights are associated with objects.
b) The primary owner can only merge the use-rights for which they have the meta-rights.
c) Merge $_{Use}$ gives use-rights, but meta-rights remain with the primary owner.
d) Consent of both the primary owner and the beneficiary is required for use-rights merge.
e) The primary owner can merge some/all of their use-rights with the joint beneficiary.
f) The primary owner can merge same right over an object with multiple joint beneficiaries at the same time.
g) The primary owner cannot exercise the merged use-right without the consent of the joint beneficiary.
h) The joint beneficiary cannot further give away the use-right.
i) The rights of primary owner and joint beneficiary are merged so they need to act jointly, but the rights of parent, offspring and general public role remain the same after merging the use-right.

5.3.6 Revocation

Merging use-rights is a common practice in collaborative software environment as multiple actors work together to achieve a common goal. However rights allocations as well as team memberships are not always static so it requires a mechanism to revoke the merged rights (Barka, 2002; Crampton & Khambhammettu, 2006, 2008; Ruan & Varadharajan, 2010). In the running example, *Alice* – the owner of the document, may assign edit rights to *Bob* over her document for a certain period of time, which may expire or change depending on the type of user contract. So the rights over 'Thesis' can be assigned to *Bob* for say two months[81], and after that it may get expired. This section covers the possible ways in which a merged right can be revoked from the joint beneficiary. As the primary owner retains the meta-rights, *Alice* has the authority to take back the editing right over her document from *Bob* (or the whole community) by revoking the right or by deleting the document at any time.

A right allocation is revocable if the initiating party keeps the meta-rights and when the Merge $_{Use}$ model allocates use-rights to the beneficiary, the meta-rights remain with the owner, so the rights are revocable. This gives the principle:

P.5.3.6: The primary owner can revoke the merged right at any time.

There are two types of revocation supported in the Merge $_{Use}$ model, that is, self-revocation and time based revocation. Rule based revocation is not needed as the primary owner has the right to view the changes done by the joint beneficiary before accepting it and only allows them if they do not violate the rules.

5.3.6.1 Self revocation

The owner revokes the use-rights based on inappropriate use, difficulty in coordination or at will. In the above scenario, *Alice* can revoke the edit rights over her document from *Bob* at any time at will. Self-revocation gives the sense of authority to the owner that the object still belongs to them even after merging rights over it with other actors.

[81] Or for the lifetime of the team.

Allowing self-revocation renders easy retrieval of the use-rights by the owner when there is a deadlock between the joint beneficiaries. It also maintains the collaborative sense between the joint beneficiaries so they may not act too extremely to refuse all suggestions.

5.3.6.2 Time based revocation

At the time of merging the use-rights, the owner can assign a time-stamp with the assigned right for the proposed lifespan of the allocation. When the time-stamp expires, the allocation is automatically revoked from the joint beneficiary. Time based revocation provides a tool that the owner can utilize to preset the revocation time for each right for each object when the team separates. In the above scenario, *Alice* can set a time-stamp on the right over the document so it is revoked after the team contract expires. Time based revocation provides the facility to revoke the allocations in the future without additional involvement of the owner.

5.3.6.3 The Merge $_{Use}$ revoke process

The revocation removes the use-rights from the joint beneficiary $JBen_u$ and gives the full rights back to the primary owner, if no other joint beneficiary is left for the particular right over the object.

$$\begin{matrix} \neg UR \rightarrow Ben_u^i \\ UR \rightarrow PO_i \end{matrix} \forall\, UR \rightarrow O \ldots [eq.\,5.38]$$

It also takes back the $JBen_u$ role from VU_j and PO_i role from SH_i.

$$\begin{matrix} \neg\, JBen_u^i \rightarrow VU_j \\ \neg\, PO_i \rightarrow SH_i \end{matrix} \forall JBen_u^i, PO_i \in LR \wedge VU_j, SH_i \in NS_i \ldots [eq.\,5.39]$$

5.3.7 Summary of Merge $_{Use}$

To summarize, the Merge $_{Use}$ model allow the owner of an object to merge the use-rights of all the actors and their joint consent is needed to exercise that right. The characteristics are given in section 5.3.1, which distinguish the model from other models of use-rights allocation. Section 5.3.2 explains the allocation process, followed by the logical definition of the model in section 5.3.3. Section 5.3.4 illustrates the rights analysis, the design principles are mentioned in section 5.3.5 while the revocation is discussed in section 5.3.6.

5.4 Chapter summary

This chapter discussed the use-rights allocation for online social interactions by outlining the Replace $_{Use}$, Share $_{Use}$ and Merge $_{Use}$ models. It also outlined the revoke process for each case. The Replace $_{Use}$ model was demonstrated with the example of a conference system, the Share $_{Use}$ model was explained using the Video On Demand system, while the Merge $_{Use}$ model was illustrated with a collaborative software example.

Chapter 6

Meta-Rights Model

This chapter establishes the basic understanding behind the allocation of meta-rights in online social interactions. Meta-rights are system permissions for actors to apply operations on rights[82] (Dewan & Shen, 1998; Indratmo & Vassileva, 2007; Mattas et al., 2006). As discussed in chapter 3, there are four possible options to allocate meta-rights and this chapter outlines these models in detail. The models introduced are Replace $_{Meta}$, Share $_{Meta}$, Merge $_{Meta}$ and Revoke $_{Meta}$. However, as the revoke process is different for each of the three allocation models, it is discussed under each allocation model section rather than in a separate revoke model section. Every model is described by using four distinct steps: a) the general description of the model along with various rights allocation characteristics and their values for online social interactions, b) the mapping of various components of SAC model to describe the working of the model in an online social interaction instance, c) the logical definition of the model to illustrate the exact possible combinations of rights over an object for the roles of owner and the beneficiaries, and d) the rights analysis of different roles of parent, offspring and general public associated with the object. After outlining each allocation model, its revocation process is discussed. The meta-rights models are usually associated with ownership

[82] They allow the owner to grant use-rights to other actors.

and administration of objects and thus Replace $_{Meta}$ often termed as object transfer, Share $_{Meta}$ is termed as secondary ownership while Merge $_{Meta}$ is termed as joint ownership.

6.1 Replace $_{Meta}$ model

Replace $_{Meta}$ model allows the owner to replace the actor for meta-rights[83], and is commonly termed as transfer (Crampton & Khambhammettu, 2006, 2008). This is done by changing the actor entity for all meta-rights triplets[84]. The need of rights transfer arises when the owner of an object wishes to permanently transfer it to another actor (Barka, 2000), who becomes the new owner. Transfer gives away all entity meta-rights (Gaaloul et al., 2008), so the previous owner loses the owner role for the object, and so cannot exercise any right over the transferred object. Also the beneficiary becomes the new owner, who is completely responsible for the object and all rights associated with it. Rights are irrevocably given to the new owner, for example after selling a house the old owner has no right to it and so cannot take it back. Transfer is used in scenarios when the owner no longer wishes to maintain the object, or the object rightfully belongs to the other actor, so the owner transfers the object and gives all its privileges to the new owner[85]. In the physical world, selling/purchasing of house/car is an example of rights transfer. In these examples, the purchaser gets all the rights over the object including the ownership and the seller remains with no right.

Let's consider the scenario of the conference system which was discussed in the previous chapter[86], but this time the focus is on the copyright of an accepted paper. The conference chair is the owner of the conference space which is also the parent space for submitted papers, and authors are the owners of papers. Many research papers are submitted to a conference and undergo a peer review process (Whitworth & Friedman, 2009a, 2009b). Reviewers often judge them on the basis of quality, relevance, novelty and presentation and give their expert opinion to conference chair to decide the accept/reject outcome. If some paper is accepted, the authors are required to submit the final draft after incorporating the reviews, and a copyright

[83] To make them the new owner.

[84] The new actor then can add themselves for use rights as well.

[85] The parent role is notified about the transfer of the child object.

[86] Chapter 5, section 5.1 (Replace $_{Use}$ model).

form to transfer the use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive rights over the paper to the conference/organization.

To map this scenario on the core SAC model, the conference system is the parent space for all the conferences owned by the system administrator. The conference is the namespace with conference chair as the stakeholder, different papers/comments as objects, and various local roles are associated with each conference. The parent has the right to view/delete any paper, which is rejected or not matched to the conference theme for example, while the owner (author) has the right to decide the copyright transfer.

Now suppose that *Alice* is the conference chair of a conference the *'Security Conference'* and *Bob* has a paper accepted in her conference. At the time of final camera ready submission, in order to publish the paper, *Bob* is required to sign a copyright form to transfer the display right[87] over his research paper to *Alice* and give her the authority over it. *Alice* wants full authority over the research paper as she has over her other objects – so she can publish or reproduce the work without concerning any other authority. She also wants that she (or the organization she is working for) should be able to decide who can read that research or who cannot[88]. So she wants that *Bob* should not only transfer all the use-rights but also the meta-rights over the paper to handle those rights. *Bob*, in order to publish his work and to get the credit[89], needs to give that authority to *Alice* (or her organization)[90]. In this case, *Bob* chooses to transfer the publish rights over his paper to *Alice*, including the meta-rights. This scenario is depicted in figure 6.1, where *Bob* – the original owner of *'paper123'* has transferred his paper to *Alice* –who will manage the view rights over it.

6.1.1 Characteristics of Replace $_{Meta}$

Following are some of the characteristics of this Replace $_{Meta}$ model that distinguish it from other models. Some of these characteristics are defined in the literature (Barka & Sandhu, 2000b; L. Zhang et al., 2003; X. Zhang et al., 2003) for role to role membership transfer in RBAC (Ferraiolo & Kuhn, 1992; Ferraiolo et al., 2001; Sandhu et al., 1996). However, the

[87] The meta-right to view.

[88] May be based on some subscription fee or affiliation.

[89] In the form of publication count and recognition, which also leads to promotions and grants.

[90] Under certain conditions, of course, like the credit should always go to *Bob* and the work will never be edited.

values of these parameters are different in this research from the values proposed in the literature due to the nature of online social interactions and the ownership domain.

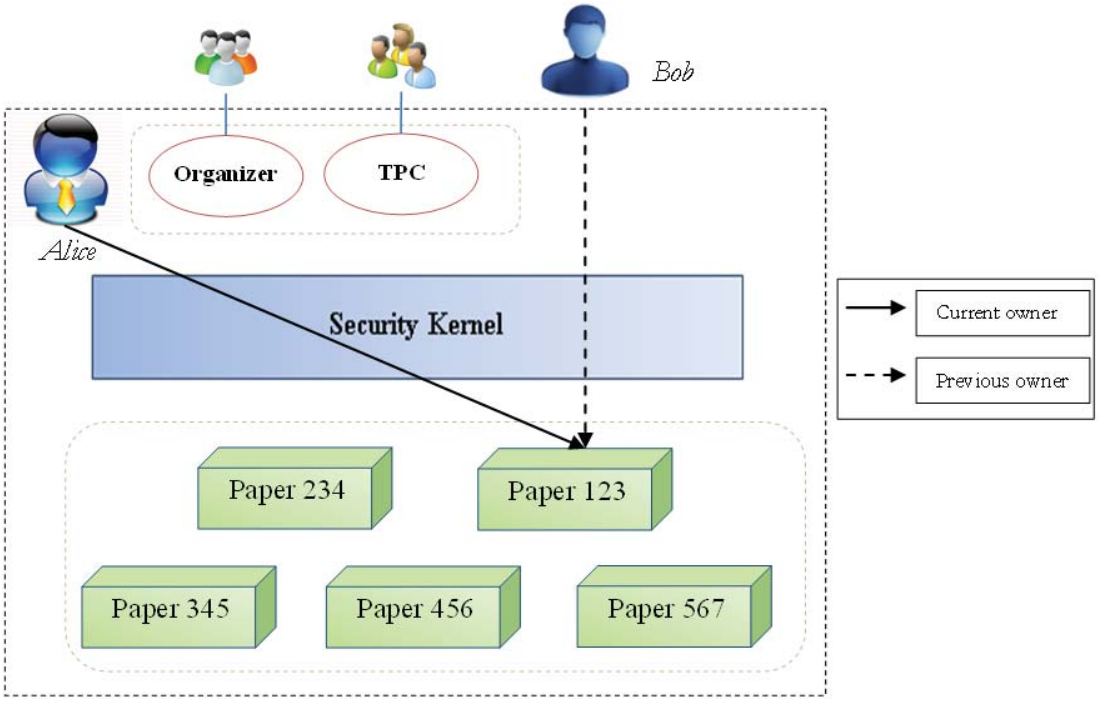

Figure 6.1: Transfer scenario depicting copyright of accepted paper

There are two basic axioms for this model, which are kept consistent throughout all the other meta-rights models in this research. First, the model presented in this research is domain based as rights are associated with objects, and second, that no one can merge a right associated with an object which they do not own. So, it is not possible for *Bob* to transfer *Alice* some other paper *'Paper234'*, where he does not have any meta-right.

6.1.1.1 Consent

As the paper is owned by *Bob* so only he can decide whether he wants to publish it in this conference or not based on his personal discretion, which gives that the system should allow the owner to decide about the transfer of the object. Also, papers accepted in one's conference establish the repute of the conference chair. It gives that *Alice* would be known for the credibility and quality of accepted papers in her conference, which could affect her online and offline reputation in her social circle. For example, she may not like to accept the transfer of a paper, which was rejected in the review phase. So *Alice* should be asked before transfer of any object to her, in this case acceptance decision before the actual transfer of the paper.

The above implies that the rights allocation system should support users' trust and provide the guarantee that no object will be transferred without the consent of its owner. Also, as objects can contribute to one's repute, the new owner should also be agreed to take all the responsibilities associated with the object. However, the allocation contract is not used for the transfer of the object as it cannot be revoked from the new owner. It gives the operational principle:

P.6.1.1: Consent of the previous and new owners is required for transfer of an object.

6.1.1.2 Totality

Totality deals with allocating the proper/improper subset of rights over proper/improper subset of authorized resources. In the above example, as the paper belongs to *Bob,* it is up to him that how much meta-rights he wants to replace with *Alice*. He may want to replace all the rights over the paper with *Alice*, in which case the replace is total, or he may want to replace only some of the meta-rights with *Alice* but keeping rest with him, in which case the replace is partial. The model supports replacing partial set of meta-rights, where *Bob* may sign the copyright and transfer the view meta-right over the paper to *Alice* but he may want to keep the meta-right to edit or delete it.

Generalizing the above scenario gives that the $\text{Replace}_{\text{Meta}}$ model deals with partial set of meta-rights as it has more significance for online social interactions. Also, total transfer of an object[91] can be achieved through multiple-partial transfers, which gives

$$Rep_m(MR_{all}) = Rep_m(MR_1) \wedge Rep_m(MR_2) \wedge \ldots \wedge Rep_m(MR_n) \ldots [eq.\ 6.1]$$

However, the replaced meta-right cannot not be taken back from the beneficiary as it cannot be exercised by the old owner. Transfer is the permanent replacement of meta-rights so the model supports partial transfers, which gives the following design principle:

P.6.1.2: The owner can transfer proper/improper subset of their rights.

[91] All the meta-rights over it.

6.1.1.3 Cardinality

Cardinality deals with the number of beneficiaries in a particular rights allocation operation. In the copyright example, *Bob* was the owner of the research paper and he transferred some rights over it to *Alice*. This gives

$$Rep_m \rightarrow (MR_i)\{Alice \mid Alice \in VU \forall VU \in NS_i\} \ldots [eq.6.2]$$

He cannot transfer the copyright to more than one conference at the same time, which may encourage corruption in STS, for example in the form of multiple publication of the same paper. However, if *Alice* wants to have more than one owner over the object, she can later use the Share $_{Meta}$ model to do it.

Generalizing the above gives that the right can only be transferred to a single actor, however, if the right needs to be transferred to more than one actor, it can be done by the new owner after the transfer operation takes place[92]. This singular cardinality eliminates the corruption at rights level, so one cannot transfer one object to multiple recipients at the same time[93]. The possibilities that are associated with giving the ownership to multiple beneficiaries are left for the case of Share $_{Meta}$, which is covered later[94] . The benefit of restricting the cardinality is the simplicity to maintain the accountability of atomic operation associated with the new and old owners, and precedence of various allocations over each other.

This gives the design principle:

P.6.1.3: It is not possible to transfer meta-rights to more than one beneficiary.

6.1.1.4 Monotonicity

Monotonicity refers to whether the previous owner can exercise the right after the allocation. In the running example, if *Bob* transfers the meta-rights over the paper to *Alice*, ownership and accountability principles demand that he cannot manage the object anymore, but only *Alice* will

[92] When he becomes the new owner.

[93] The traditional models presented in literature works on multiple cardinality of rights, while the cardinality of transfer is singular in offline communities and so this model supports the singular cardinality following socio-technical design.

[94] Section 6.2.

do any management. This condition is important as the need of transfer arises because of the demand of new owner to manage the object. If either *Alice* cannot manage the access of other actors over the paper, or *Bob* can still allow others to access it, the purpose of meta-rights transfer is not fulfilled. It is also justified in this case as the transfer of copyright comes at the cost of some other benefit – publication and recognition in this case[95] which *Bob* will enjoy[96].

Replace $_{Meta}$ model replaces the actor in the owner role for the right and gives the rights of old and new owner by

$$Rep_m\ (OldOwner) \rightarrow \begin{matrix} UR(OldOwner, O, \phi) \\ MR(OldOwner, O, \phi) \end{matrix} \ldots [eq.6.3]$$

and

$$Rep_m\ (NewOwner) \rightarrow \begin{matrix} UR(NewOwner, O, All) \\ MR(NewOwner, O, All) \end{matrix} \ldots [eq.6.4]$$

The above discussion can be generalized that transfer is mutual exclusive, that is, if meta-rights over an object are transferred they cannot be further exercised by the old owner, but only the new owner can exercise those rights[97]. Even if the object is not handled very well by the new owner, the old owner has no right over it, for example after selling the house the old owner remains with no right over it. This gives the design principle:

P.6.1.4: After the transfer, the old owner has no right over the object.

6.1.1.5 Depth

Depth deals with allowing the beneficiary to further pass on the allocated right. In the proposed model, if the new owner gets all the meta-rights over the object, they can again modify the meta-rights triplet to pass the object to some other actor, for example selling of a second hand car. The Replace $_{Meta}$ model supports multi-level depth[98] of meta-rights transfer,

[95] Which may lead to other indirect benefits of promotions, grants and so on.

[96] In physical world, transfer also gives some benefits to the old owner in form of money or exchange with other needs and so on.

[97] The new owner can assign some rights to old owner as they can assign rights to other actors.

[98] The old owner can restrict this by keeping some of the rights using Merge $_{Meta}$ model (Section 6.3).

which supports ownership and makes the model recursive, where one can transfer any right after holding the meta-right.

In the presented scenario, *Alice* can pass the paper to any other actor or organization as she has the meta-rights over it. *Bob* cannot restrict *Alice* to do any further transfers as he does not have any meta-right left over the paper[99]. This gives the operational principle:

P.6.1.5: Transfer allows the new owner to further transfer the object.

6.1.2 Replace $_{Meta}$ process

The Replace $_{Meta}$ model maps various components of the SAC model to each other and defines the transfer operation using the following set of rules:

a) The requestor VU_j belongs to the LR_i in the NS_i

$$VU_j \rightarrow LR_i, \ \forall \ LR_i \in NS_i \ldots [eq.6.5]$$

b) The requested object O is classified into the same OC_τ.

$$O \rightarrow OC_\tau, \forall \ OC_\tau \in NS_i \ldots [eq.6.6]$$

c) The transfer of object O from NS_i to NS_j

$$O \rightarrow NS_j, \exists \ NS_j \mid VU_j \in (SH_j, NS_j) \ldots [eq.6.7]$$

d) The replacement of VU_j with SH_i in the *Owner* role for the object

$$VU_j \rightarrow Owner_o, \forall \ O \in NS_j \ldots [eq.6.8]$$

which results in the following two operations

e) The addition of use-rights and meta-rights to the VU_j.

$$\begin{matrix} UR \rightarrow VU_j \\ MR \rightarrow VU_j \end{matrix} \forall \, UR, MR \rightarrow O \ldots [eq.6.9]$$

f) The removal of use-rights and meta-rights from the SH_i.

$$\begin{matrix} \neg UR \rightarrow SH_i \\ \neg MR \rightarrow SH_i \end{matrix} \forall \, UR, MR \rightarrow O \ldots [eq.6.10]$$

Given a *NS i*, a transfer request is only granted if the virtual user VU_j acquires a local role LR_i in namespace NS_i. Also the object O is mapped in object class OC with the privacy label τ. The

[99] He may restrict *Alice* at the time of transfer by making some conditional contract.

object is moved from NS_i to NS_j, which adds the meta-rights to VU_j and removes them from SH_i.

6.1.3 Definition

$\text{Replace}_{\text{Meta}}$ is an operation that replaces the active actor set over use-rights and meta-rights with another actor, that is, the new owner. It can be defined as follows:

NS, SH, Opr, LR, UR, MR and O are sets of Namespaces, Stakeholders, Operations, Local Roles, Use-Rights, Meta-Rights and Objects respectively. After the object has been moved from NS_i to NS_j, the following role and rights are assigned to SH_j:

- A transfer of owner role from (OldOwner) VU_i to (NewOwner) VU_j
- A many-to-many NewOwner role to all rights assignment relation is given by

$$NewOwner\_UR \subseteq NewOwner \times UR$$
$$NewOwner\_MR \subseteq NewOwner \times MR$$

- $NewOwner\_rights: (NewOwner: LR) \rightarrow 2^{UR}$ is a function derived from New*Owner*_UR, for mapping NewOwner role to a set of use-rights.

$$NewOwner\_rights\,(NewOwner) = \{NewOwner \in LR | (Newowner, ur) \in NewOwner\_UR, \forall\ LR \in NS_j\}$$

- $owner\_rights: (Owner: LR) \rightarrow 2^{MR}$ is a function derived from *NewOwner*_MR, for mapping NewOwner role to a set of meta-rights.

$$Newowner\_rights\,(NewOwner) = \{NewOwner \in LR | (Newowner, mr) \in NewOwner\_MR, \forall\ LR \in NS_j\}$$

Access Grant

The transfer model does not affect the way an object can be accessed but only the virtual user in the owner role is changed.

6.1.4 Rights analysis

This section describes the modification of rights for various roles of parent, offspring and general public associated with the transferred object. Transfer replaces the meta-rights owner

role from one actor to another, which may results in change of space in some cases[100], it also changes the rights of all the roles associated with that object.

If the object is transferred from old owner to new owner but resides in the same parent space, then only the old and new owner rights will be changed but the rights of parent, offspring and general public role remain the same. However, if the object is moved from one parent space to another parent space then the rights of old and new parent, old and new general public, and old and new owners will be changed. On the contrary, if the transferred object is a space and has some dependent objects, all the child objects will move along with their parent[101] so, offspring rights remain the same for the object being transferred in both cases.

Consider the copyright example, the transferred paper is transferred from *Bob* to *Alice*, where *Alice* was previously the owner of the parent space – conference, so the rights of parent, offspring and general public role will be changed. The rights of old owner – the paper author, reduce from all the rights to none after transferring it to *Alice* – the conference chair[102]. The new owner's rights increased from some to all – use as well as meta-rights.

Below are the tables for different rights of different old and new roles associated with object before and after the Replace $_{Meta}$ model.

	Old Parent	**Old Owner**	**Old Offspring**	**Old G. Public**
Before Rep $_{Meta}$	UR(V, D)	UR(V,D,E) MR(V,D,E)	UR(V)	D/C
After Rep $_{Meta}$	NR	NR	UR(V)	NR
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively D/C represents don't care and depends on beneficiary previous role in the space NR represents No Right				

Table 6.1 (a): Rights for different old roles associated with object before and after the Replace $_{Meta}$ model

[100] In other cases, it may happen from one author to another author for example.

[101] As change of host server in case of website does not affect the contents of the website.

[102] He may view his own copy but to view the published copy, he may need to subscribe for the organization at some cost, just like any other actor.

	New Parent	New Owner	New Offspring	New G. Public
Before Rep $_{Meta}$	D/C	D/C	UR(V)	D/C
After Rep $_{Meta}$	UR(V, D)	UR(V,D,E) MR(V,D,E)	UR(V)	D/C
V, D, E and C represents view, delete, edit and create respectively UR and MR represents use-rights and meta-rights respectively D/C represents don't care and depends on beneficiary previous role in the space				

Table 6.1 (b): Rights for different new roles associated with object before and after the Replace $_{Meta}$ model

Below is table 6.2 for the roles of old and new owner, and their rights over parent, current, and child objects.

	Old Owner			New Owner		
	Parent	**Transferred Obj.**	**Child**	**Parent**	**Transferred Obj.**	**Child**
Before Rep $_{Meta}$	UR(V)	UR(V,D,E) MR(V,D,E)	UR(V, D)	D/C	D/C	D/C
After Rep $_{Meta}$	NR	NR	NR	UR(V)	UR(V,D,E) MR(V,D,E)	UR(V,D)
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively D/C represents don't care and depends on beneficiary previous role in the space						

Table 6.2: Old and new owner rights over different objects before and after the Replace $_{Meta}$ model

A complete analysis of rights for different roles based on table 6.1 (a, b) and 6.2 for the Replace $_{Meta}$ model is as follows:

a) After Replace $_{Meta}$, the old owner remains with no right over the object. However, if he acquires some role in the new owner space, for example general public, he would only get the rights associated with that role.
b) The old owner also loses all the rights to the parent space unless he has the membership for general public role or has some other object in the same parent space.
c) The old owner remains with no right over the child object.
d) The new owner gets all the rights over the object including the meta-rights, so they can manage other actors' access.
e) The new owner gets enter/view right over the parent space.
f) The new owner also gets all the rights of parent role over the child objects.

g) If new and old owners are offspring of the same parent space, then the rights of parent will not change.

h) If the object remains in the same parent space the rights of the general public role remains the same.

i) If the object is transferred from one parent space to another[103], then the old parent and old general public role left with no rights over the object. New parent role gets all the rights appropriate for them (for example view and/or delete right) and new general public role may get some view rights (for example view) depending on the new owner's space configuration.

j) If the transferred object is a space and contains other child objects, they will also transfer with their parent space. So in both cases, the rights of offspring role will not change.

6.1.5 Design principles

The above discussion gives the following design principles for the Replace $_{Meta}$ model.

a) The model addresses domain based object transfers, where rights are associated with objects.

b) An actor can only transfer objects under their ownership and over which they have the meta-rights.

c) Transfer gives use-rights and meta-rights to the new owner.

d) Transfer should notify the parent about old and new owner.

e) The consent of old and new owner is required for any transfer.

f) An actor can transfer any subset of complete meta-rights set.

g) An actor can only transfer an object to single beneficiary.

h) Old owner cannot exercise any right over transferred object.

i) The new owner can further transfer the rights over the object.

j) The rights of old parent, old owner and old general public roles will be removed, while new parent, new owner and new general public role acquire all the rights. The rights of offspring role remain the same.

[103] The change in parent space.

6.1.6 Revocation

A right allocation is revocable if the old owner keeps the meta-rights, but transfer allocates use as well as meta-rights to the new owner, so is irrevocable. In this case, the old owner is no more responsible for the object nor can he exercise/ manage any right over the object, and the new owner can exercise all the management of object, which cannot be revoked.

In the above scenario, *Bob* remains with no authority to withdraw the paper from *Alice*'s conference after it is being published, giving the operation principle:

P.6.1.6: Transfer is irrevocable, so the old owner cannot take back the rights.

6.1.7 Summary of Replace $_{Meta}$

The difference between the characteristics of the proposed transfer model and the traditional transfer models can be seen in table 6.3.

Characteristic	Proposed Model	Traditional Model
Consent	Joint	Several
Totality	Partial	Total
Cardinality	Singular	Singular
Monotonicity	Mutual Exdusive	Mutual Exdusive
Depth	Chain	Chain
Revocation	No	Yes
Who can transfer	If one has the meta-right	Owner/Security Administrator/ If one has the use-right[104]
Who can revoke	No one	Security Administrator

Table 6.3: Difference between proposed and traditional delegation models

In role transfer, the complete role is transferred from one user to another as all the rights over all the objects are associated with roles, while the presented model treats each right separately and the owner has the option to transfer subset of rights over subset of objects.. Also, the Replace $_{Meta}$ model is based on discretionary grounds, whereas the transfer in RBAC was based on mandatory grounds and a security officer is responsible for assigning roles (Sandhu & Munawer, 1998a, 1998b, 1999b). This central administration approach cannot work for STS

[104] Member of the role.

which are based on ownership and has the potential of millions of users. Furthermore, the transfer in RBAC allows the security officer to revoke the transfer (Barka, 2002), while the model presented here not allows anybody to revoke the transfer of an object[105].

To summarize, the Replace $_{Meta}$ model allows the owner of an object to transfer the use as well as meta-rights over his object to some other actor, who becomes the new owner. The characteristics are given in section 6.1.1, which distinguish the model from other models of use-rights allocation. Section 6.1.2 explains the allocation process, followed by the logical definition of the model in section 6.1.3. Section 6.1.4 illustrates the rights analysis, the design principles are mentioned in section 6.1.5 while the revocation is discussed in section 6.1.6.

6.2 Share $_{Meta}$ model

Share $_{Meta}$ model allows the owner to allocate the meta-rights over his object to other actor while keeping the same meta-rights, so both can exercise it[106] (Y. Liu et al., 2009; Z. Zhang et al., 2008). In the meta-rights triplet this is done by adding an actor to the existing set of actors, while the use-rights triplet does not change[107]. The need of Share $_{Meta}$ arises when the owner of an object wants to give meta-rights over his object to other as well as keep them so both can manage the rights over the object. It is used in distributed scenarios where multiple actors own the same object and manage the distribution of rights over it, for example multiple organizers of the same event. The Share $_{Meta}$ model adds a new actor for meta-rights but the rights of existing actors do not change. In Share $_{Meta}$, the meta-right is given completely so any party can act alone without any involvement as if they owned it exclusively. As both the original owner and the beneficiary have the meta-rights, either of them can transfer the object or abstain the other owner from use or meta-rights[108]. In the physical world, multiple salesmen working in a store is an example of sharing *sell* meta-rights[109], where any of them can transfer the object[110].

---

[105] Only the new owner can again transfer the object back to the old owner.

[106] This is the reverse case for Replace $_{Meta}$ with respect to the owner, where the original owner loses his rights.

[107] The beneficiary gets the meta-rights so can add use-rights to them if they want.

[108] For this reason, sharing of meta-rights needs some contract which poses some restrictions on the beneficiaries.

[109] The salesmen have the right to transfer the use-rights and meta-rights over the object but they cannot use it themselves.

[110] Who becomes the new owner.

Let's consider the scenario of sharing a video on *'Facebook'*, which is a social networking service allowing users to share their information and activities with their friends and family. A wall is associated with every Facebook account which displays the activities of its owner. These activities involve reading various articles, watching videos, uploading pictures and posting comments. Users add friends on *'Facebook'* and exchange messages which also include automatic notification when one of their friends updates their wall. Every wall has different roles of friends, family and so on, among which the owner manages the access rights over his objects. Every video (or any other object) has various roles of owner, commenters, likers, viewers, sharer and general public.

To map this scenario on the core SAC model, the Facebook system is the parent space of all the profiles, walls and so on, owned by the system administrator. Wall is the parent space for all the videos posted in it, and likes and comments are child objects of the video space. Every video in the system has different set of parent, offspring and general public roles. The system space is the namespace with system administrator as the stakeholder, and wall is the namespace with profile owner as stakeholder. The profile owner has the right to create a video, display it to some users, or delete it.

Now suppose that an actor *Alice* creates a video on her wall, and wants to share it with *Bob* – an actor in her friend role. She wants that *Bob* can view the video, may comment or like it, and also can show it to his friends/family if he wants. She then shares the view meta-right with *Bob* so he can manage the view access for his friends over the video. As *Bob* has the view meta-rights, he can further share the video with *Carl* – results in forwarding him the view meta-rights. Sharing the video on their own wall gives them the meta-rights over it for their own wall, but the rights of the original owner remain the same for his/her wall. This scenario is depicted in figure 6.2.

6.2.1 Characteristics of Share $_{\text{Meta}}$

Following are some of the characteristics of this Share $_{\text{Meta}}$ model that distinguish it from other models of meta-rights. Some of these characteristics are defined in literature (Barka & Sandhu, 2000b; L. Zhang et al., 2003; X. Zhang et al., 2003) for role to role membership transfer in RBAC (Ferraiolo & Kuhn, 1992; Ferraiolo et al., 2001; Sandhu et al., 1996). However, the

values of these parameters are different in this research from the values proposed in the literature due to the nature of online social interactions and the ownership domain.

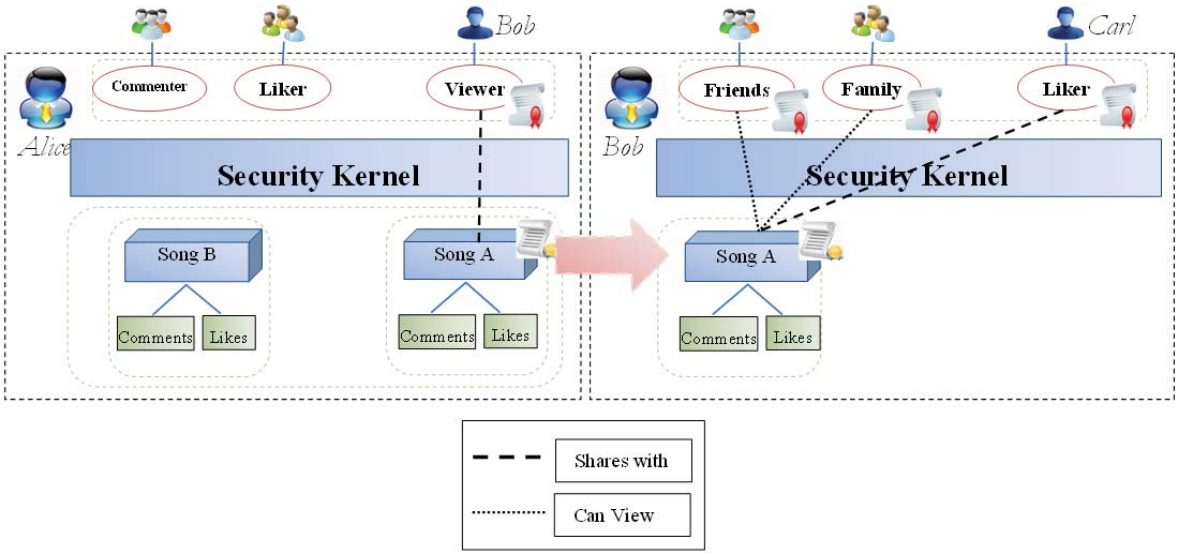

Figure 6.2: Sharing meta-rights scenario depicting Facebook wall

There are two basic axioms for this model, which are kept consistent throughout all the other meta-rights models in this research. First, the model presented in this research is domain based as rights are associated with objects, and second, that no one can share a right associated with an object which they do not own. So, it is not possible for *Alice* to assign *Bob* some rights over another video, where she does not have any meta-right.

6.2.1.1 Consent

Allocating the meta-rights over a video to *Bob* requires *Alice* to initiate the process as the system should enhance the owner's trust. *Alice* may want to add or delete video at will and decide whether it is appropriate to share the meta-rights over the video to a specific actor or not. Also, it is necessary to take *Bob's* consent as he would be responsible for its propagation and handling of rights over the video. Also, he must be aware of the object due to reputation management. If this condition is relaxed, one can share indecent videos with your name on your wall without your consent which is socially inacceptable.

The above implies that meta-rights are associated with one's repute[111] and gives the responsibility in some cases as well. So, meta-rights can only be shared with the consent of the owner and the beneficiary. It gives the following design principle:

P.6.2.1: Meta-rights sharing require the consent of the owner and the beneficiary.

6.2.1.2 Totality

Totality deals with allocating the proper/improper subset of rights over proper/improper subset of authorized resources. In the above example, it depends on *Alice* discretion that how much meta-rights she wants to share with *Bob* from her complete set of meta-rights. She may give him only the view meta-rights so he can only manage the users to view the video, in which case the sharing is partial, or she may give the edit/delete meta-rights so he can allow others to edit/delete the video, in which case the sharing is total. Total sharing of meta-rights creates another primary owner with the same rights over the object and may not be appropriate in all conditions, so this Share $_{Meta}$ model supports partial sharing. Also, partial can be extended to the complete meta-right set, in cases where complete set of meta-rights need to be shared using

$$Shr_m(MR_{all}) = Shr_m(MR_1) \wedge Shr_m(MR_2) \wedge \ldots \wedge Shr_m(MR_n) \ldots [eq.\ 6.11]$$

Generalizing the above implies that there are two possible design options in the Share $_{Meta}$ model, that is, only complete sharing of meta-rights is supported as all the meta-rights or none can be shared with the other actor, or partial sharing is supported so a subset of all the meta-rights held by the owner can be shared. This model supports partial sharing as it has more useful scenarios and also total can be achieved using multiple partial sharings in cases where it is needed. This gives the design principle:

P.6.2.2: The owner can share the proper/improper subset of their meta-rights.

[111] Having view right over an indecent/controversial video on YouTube may not affect ones repute as they are part of general public role. However, having the same video on ones wall with their name may be of more concern.

6.2.1.3 Cardinality

Cardinality deals with the number of beneficiaries in a particular rights allocation operation. In the above example, *Alice* can share the view meta-right over her video to multiple actors say *Bob* and *Carl* at the same time by sharing it with both of them. Now if both of them want to share it with their social circles, they can do it because they have the view meta-right over the object, which gives

$$Shr_m \rightarrow (MR_i)\{(Bob \mid Carl) \mid Bob, Carl \in VU \; \forall \; VU \in NS_i\} \ldots [eq.\ 6.12]$$

Generalizing the above case gives that the same meta-right over the same object can be shared with multiple beneficiaries, which gives the operational principle:

P.6.2.3: It is possible to share same meta-right to more than one beneficiary.

6.2.1.4 Monotonicity

Monotonicity refers to whether the previous owner can exercise the right after the allocation. In the running example, if *Alice* shares view meta-right over her video to *Bob*, she can still manage the view access over the video, that is, can exercise the meta-rights. The case where the initiating party cannot exercise the right is covered in the Replace $_{Meta}$ model[112] and so this Share $_{Meta}$ model covers the cases where the owner and the beneficiary both can exercise the right at the same time[113].

The Share $_{Meta}$ model adds the beneficiary to the actor set for meta-right, which gives

$$Shr_m \rightarrow MR(\{Alice|Bob\}, O, opr) \ldots [eq.\ 6.13]$$

Share $_{Meta}$ creates two roles with each shared meta-right associated with the object, that is, Primary Owner *(PO)* and the Secondary Owner *(SO)*. It means that given *x* rights associated with an object, the Share $_{Meta}$ model can create up to *2x* roles, which is convenient for assigning rights to roles as well. For actor centric implementations, the rights associated with them can be considered as

[112] Section 6.1.

[113] The case when both of them need to jointly agreed upon some operation is covered in Merge $_{Meta}$ (section 6.3).

$$Shr_m(SO) \rightarrow \begin{matrix} UR(\{SO\}, O, opr) \\ MR(\{SO\}, O, opr) \end{matrix} \dots [eq.6.14]$$

and

$$Shr_u(PO) \rightarrow \begin{matrix} UR(\{PO\}, O, opr) \\ MR(\{PO\}, O, All) \end{matrix} \dots [eq.6.15]$$

It can be generalized that the sharing of meta-rights is mutual inclusive, so if some meta-right is shared between two actors, then both can exercise it as they have the complete right. This provides the sharing of authority and opportunities for team work, as one can share meta-rights with another actor and both of them can accomplish the task. This gives the design principle:

P.6.2.4: Both the primary and secondary owner can exercise the shared meta-right.

6.2.1.5 Depth

Depth deals with allowing the beneficiary to further pass on the allocated right. In the proposed model, as the secondary owner gets meta-rights, they can pass the use or meta-rights to others. Sharing of meta-rights addressed in this model is multistep, so the beneficiary can further pass the right to others. This condition permits the sharing of information, pictures and videos on social networks, and allows the information to propagate from within a single social circle to the whole community.

In the running example, *Alice* may share the view meta-rights to *Bob*, who can further share it with other actors, for example *Carl*. The social circles of *Bob* and *Carl* can further share the video with their social circle if *Bob* and *Carl* have shared the meta-rights with them, which can be seen by

$$Shr_m \rightarrow MR(\{PO|SO\}, O, opr) \dots [eq.6.16]$$

This condition is important for system growth and gives the operational principle:

P.6.2.5: Sharing meta-rights allows the beneficiary to further share it.

6.2.2 Share $_{Meta}$ process

The Share $_{Meta}$ model maps various components of the SAC model to each other and defines the share operation using the following set of rules:

a) The VU_j belongs to the LR_i in the NS_i

$$VU_j \rightarrow LR_i, \ \forall \ LR_i \in NS_i \ldots [eq.\ 6.17]$$

b) The requested object O is classified into the same OC_τ

$$O \rightarrow OC_\tau, \forall \ OC_\tau \in NS_i \ldots [eq.\ 6.18]$$

c) The assignment of SO_i role to VU_j, and PO_i to SH_i

$$\begin{matrix} SH_i \rightarrow PO_i \\ VU_j \rightarrow SO_i \end{matrix} \forall \ PO_i, SO_i \in LR_i \ldots [eq.\ 6.19]$$

d) The sharing of use-rights and meta-rights with the SO_i

$$Shr_m \rightarrow \begin{matrix} UR \rightarrow SO_i \\ MR \rightarrow SO_i \end{matrix} \forall\, UR, MR \rightarrow O \ldots [eq.\ 6.20]$$

e) The rights for PO_i remain the same.

$$\begin{matrix} UR \rightarrow PO_i \\ MR \rightarrow PO_i \end{matrix} \forall\, UR, MR \rightarrow O \ldots [eq.\ 6.21]$$

Given a $NS\,i$, a share request is only granted if the virtual user VU_j acquires a local role LR_i in namespace NS_i. Also the object O is mapped in object class OC with the privacy label τ. Further, VU_j is assigned to the secondary owner role SO, which has the meta-rights over the object O.

6.2.3 Definition

Share $_{Meta}$ is an operation that adds an actor to the actor set for meta-rights. It can be defined as follows:

NS, SH, VU, Opr, LR, PO, SO, MR, and O are set of Namespaces, Stakeholders, Virtual Users, Operations, Local Roles, Primary Owner, Secondary Owner, Meta-Rights and Objects respectively.

- A one-to-many stakeholder to Primary Owner role assignment relation is given by

$$SH\_PO \subseteq SH \times PO$$

- $assigned\_PO: (sh: SH) \rightarrow 2^{PO}$ is a function derived from SH_PO, for mapping of

primary owner role onto stakeholder, which can be ϕ (empty) or the SH for the Share $_{Meta}$ operation based on the current state of the object.

$$assigned\_PO(sh) = \{sh \in SH, po \in PO | (sh, po) \in SH\_PO, \forall LR \in NS_i\}$$

- A many-to-many virtual user to Secondary Owner role assignment relation is given by

$$VU\_SO \subseteq VU \times SO$$

- $assigned\_SO: (so: LR) \rightarrow 2^{VU}$ is a function derived from VU_SO, for mapping of SO role over object to a set of virtual users.

$$assigned_{SO}(so) = \{vu \in VU, so \in SO | (vu, so) \in VU_{SO}, \forall VU, LR \in NS_i\}$$

- $AC_m = 2^{(opr \times UR)}$ is the set of attestation certificates for meta-rights.
- A many-to-many PO role to attestation certificate assignment relation is given by

$$PO\_AC_m \subseteq PO \times AC_m$$

- $PO\_rights: (po: LR) \rightarrow 2^{AC_m}$ is a function derived from PO_AC_m, for mapping each Primary Owner role to a set of attestation certificates.

$$PO\_rights\ (PO) = \{ac \in AC_m, po \in LR | (po, ac) \in PO\_AC_m, \forall\ LR \in NS_i\}$$

- A many-to-many SO role to attestation certificate assignment relation is given by

$$SO\_AC_m \subseteq SO \times AC_m$$

- $SO\_rights: (so: LR) \rightarrow 2^{AC_m}$ is a function derived from SO_AC_m for mapping each Secondary Owner role to a set of attestation certificates.

$$SO\_rights\ (SO) = \{so \in LR | (so, ac) \in SO\_AC_m, \forall\ LR \in NS_i\}$$

As all the meta-rights over an object can be distributed among one or more secondary owners, it gives the concept of active users over an object state, which can be calculated by

$$Active\ Users = PO \cup SO$$

Access Grant

An actor can perform an operation *opr* over an object *O*, under the Share $_{Meta}$ model only if there exist a mapping of vu_i for the SO role over the object assigned by the primary owner, and the SO role is authorized for the requested meta-right *mr*, so making vu_i a member of active user set, under the following relation:

$$vu : VU, AU: Active\ Users, ac: AC_m \forall VU, AU \in NS_i$$

$$vu \in assigned\_SO\ (vu) \wedge ac \in SO\_rights(so) \Longrightarrow vu \in AU$$

And the beneficiary of a use-right over a shared meta-right is the union of all the actors

assigned by all the primary and secondary owners. So, an actor can perform an operation *opr* over an object *O* only if there exist a mapping of vu_i for the Ben_x role over the object assigned by the primary or secondary owner, and the Ben_x role is authorized for the requested use-right *ur*, so making vu_i a member of allowed user set, under the following relation:

$$vu : VU, \quad AU: Active\ Users, \quad ur: Use - Rights\ \forall\ VU, AU, Ben_u \in NS_i$$

$$vu \in assigned\_SO\ (so) \rightarrow AU\ \wedge\ ac \in SO\_rights(so) \Longrightarrow Access\ Granted$$

6.2.4 Rights analysis

This section describes the modification of rights for various roles of parent, offspring and general public associated with the shared object. Consider the *'Facebook'* wall example, the shared video is not moved from one namespace to another, but remain in the same space – wall, so the rights of its old owner, parent, offspring and general public role remain the same. The secondary owner's rights are increased from no meta-right to some subset, which modifies the rights of all the new roles associated with his video.

Below is table 6.4 for different rights of different roles associated with the object before and after the Share $_{Meta}$ model.

	Parent	**PO**	**SO**	**Offspring**	**G. Public**
Before Shr$_{Meta}$	UR (V,D)	UR(V,D,E) MR(V,D,E)	D/C	UR (V)	D/C
After Shr$_{Meta}$	UR (V,D)	R(V,D,E) MR(V,D,E)	R(V,D,E) MR(V,D,E)	UR (V)	D/C
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively PO represents primary owner, while SO represents secondary owner D/C represents don't care and depends on beneficiary previous role in the space					

Table 6.4: Rights for different roles associated with the object before and after the Share $_{Meta}$ model

Below is table 6.5 for the roles of primary and secondary owner, and their rights over parent, current, and child objects.

	PO			SO		
	Parent	**Shr. Obj**	**Child**	**Parent**	**Shr. Obj**	**Child**
Before Shr_{Meta}	UR (V)	UR(V,D,E) MR(V,D,E)	UR (V,D)	D/C	D/C	D/C
After Shr_{Meta}	UR (V)	UR(V,D,E) MR(V,D,E)	UR (V,D)	UR (V)	UR(V,D,E) MR(V,D,E)	UR (V,D)
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively PO represents primary owner, while SO represents secondary owner D/C represents don't care and depends on beneficiary previous role in the space						

Table 6.5: Primary and Secondary Owner rights over different objects before and after the $Share_{Meta}$ model

A complete analysis of rights for different roles based on table 6.4 and 6.5 for $Share_{Meta}$ model is as follows:

a) After $Share_{Meta}$, the primary owner can exercise all the right over the object without reference to any other actor in the system, as before. They can also revoke the meta-rights from the secondary owner at any time.
b) The rights of primary owner over the parent space remain the same, that is, they can enter the parent space.
c) The rights of primary owner over child objects remain the same, that is, they can view and delete the child object.
d) The secondary owner rights are increased from no meta-right to some subset of meta-rights[114] over the object.
e) The secondary owner gets the view right over the parent space to enter.
f) The secondary owner gets the shared right over the child objects.
g) $Share_{Meta}$ does not change the space of the object, so parent's rights over the object remains the same, that is, they can still view and delete the object.
h) $Share_{Meta}$ model does not affect the child objects so their rights remain the same over the object, that is, they can still enter the space.

[114] Depending on the shared subset.

i) General public role is not affected by the sharing of rights over the object so their rights over the object remain the same, which depends on the system and the space configuration.

6.2.5 Design principles

The above discussion gives the following design principles for the Share $_{Meta}$ model.

a) The Share $_{Meta}$ model addresses domain based sharing, where rights are associated with objects.
b) The owner can only share the meta-rights which they have themselves.
c) Share $_{Meta}$ gives only meta-rights, but the secondary owner can add use-rights himself.
d) Share $_{Meta}$ requires the consent of the primary as well as secondary owner.
e) The primary owner can share proper/improper subsets of their meta-rights.
f) The primary owner can share same meta-right over the same object to multiple secondary owners.
g) The primary and secondary owner can exercise the meta-right at the same time.
h) As the secondary owner gets the meta-right, they can further allocate the use or meta-right.
i) The rights of primary owner, parent, offspring and general public roles remain the same after sharing the meta-right with the secondary owner.

6.2.6 Revocation

As the primary owner keeps the meta-right, they can revoke the right from the secondary owner giving that sharing of meta-rights is revocable. However, as the secondary owner also has the meta-right, they can also remove the primary owner from use or meta-rights. This condition can be restricted by making a shadow of the original object and assigning the meta-rights over it[115]. If this model is not designed carefully it may result in sharing all meta-rights among primary and secondary owners, which leads to loss of ownership and makes the system unstable. In the above example, *Alice* – the owner of the video, may assign view meta-right to

[115] Like currently done by Facebook, where secondary owner cannot delete the original video.

Bob over a shadow of the original video, so he may manage the access of his social circle without interfering the other actors in the system.

The model only supports self-revocation as time-based revocation and rule based revocation can be modified by the secondary owner after taking the meta-right. The model allows the owner to revoke the meta-right at will, but it also allows the secondary owner to revoke the rights of the primary owner. This gives the following design principle:

P.6.2.6: The meta-rights owner can revoke the shared meta-right.

6.2.6.1 Self revocation

The owner revokes the meta-right based on inappropriate use or their will. In the above scenario, *Alice* can revoke the view meta-rights over her video from *Bob* at any time at will, which results in removal of video from his wall. It is also consistent with the basic access control structure as the parent can delete the video of their offspring. Self-revocation gives the sense of authority to the owner that the object can be taken back at any time.

6.2.6.2 The Share $_{Meta}$ revoke process

The revocation removes the meta-rights from the secondary owner *SO*.

$$\neg MR \rightarrow SO \,, \forall\, MR \rightarrow O \ldots [eq.\,6.22]$$

However he may exercise the use-rights over the object under some other role. It also takes back the *SO* role from VU_i.

$$\neg\, SO_i \rightarrow VU_i \,, |\, SO_i \in LR \,\wedge\, VU_i \in NS_i \ \ldots [eq.\,6.23]$$

6.2.7 Summary of Share $_{Meta}$

To summarize, the Share $_{Meta}$ model allow the owner of an object to give the meta-rights to some other actor and keep it at the same time. The characteristics are given in section 6.2.1, which distinguish the model from other models of use-rights allocation. Section 6.2.2 explains the allocation process, followed by the logical definition of the model in section 6.2.3. Section 6.2.4 illustrates the rights analysis, the design principles are mentioned in section 6.2.5 while the revocation is discussed in section 6.2.6.

6.3 Merge $_{Meta}$ model

Merge $_{Meta}$ model allows the owner to merge the meta-rights over his object with another actor, where both must agree to manage the access rights for others. In the meta-rights triplet it is done by adding an actor to the existing set of actors and then merging the right of all of them. It allows the beneficiary to access the meta-rights triplet, but they can only modify it with the help of other meta-rights owners. The need of Merge $_{Meta}$ arises when the owner wants to give meta-rights over his object to other actors and at the same time wants to maintain his authority. In Merge $_{Meta}$, the meta-right is divided completely so no participant can act alone but anyone can stop an act which gives that all must agree to change any right. As the meta-rights are merged, it cannot be revoked by the primary owner alone but only if all the meta-rights owners agree to take the right back from the beneficiary including themselves. In the physical world, a couple who jointly own a house is an example of merging the meta-rights where both must agree to sell it. In this example, the rights of both the owners are merged to sale the house so they need to cooperate for any modification on use-rights and/or meta-rights.

Let's consider the scenario of *'Internet forum'*[116] which are online discussion boards to allow people to discuss various topics in the form of posted messages – threads. People use them to post/read information about some specific interest as they are built around group discussions from the community to the community. They provide the facility to allow single user to post a message to all the members of the forum. However for keeping the group discussion on the interest track and to avoid spam, some moderated forums require the consent of the moderator before posting a thread on it. For this, the meta-rights of the sender and the forum moderator are merged for the display of the thread. Only after the confirmation of the sender[117] and the moderator, the thread is displayed to the whole community. In case of disagreement the post is rejected and sends back to the sender[118].

---

[116] Also known as message board.

[117] In form of posting the thread.

[118] The system cannot use Share $_{Meta}$ as it will allow the sender to bypass the moderator, also it cannot use Replace $_{Meta}$ as then the moderator can modify any post or use their own name to post it.

To map this scenario on the core SAC model, the *'Internet forum'* system is the parent space of all the subgroups[119] and topics with system administrator as the stakeholder. Subgroup is the namespace with moderator as the stakeholder and message posts and their replies are child objects of the subgroup space. Every post in a moderated group has a different creator – the sender, but all are at the same hierarchical level with same role of parent and general public. The owner has the right to start a discussion, create a thread, decide whether to post the thread to this group or not, and in affect will decide who can have access to their object.

Now suppose that an actor *Alice* is the moderator of a forum '*Social Gaming*' and *Bob* is one of the users in it. *Alice* wants that if *Bob* post some thread to the forum, it may not be forwarded to the community without her consent as the community may leave due to indecent threads which can also ruin her reputation. She then merges the meta-rights of the sender and herself, for all the threads posted on her forum. If then *Bob* posts some thread '*importance of user interfaces in social gaming*' to share it with the community, *Alice's* consent is required and it would only be displayed if she finds it appropriate for the forum audience. This scenario is depicted in figure 6.3.

6.3.1 Characteristics of Merge $_{Meta}$

Following are some of the characteristics of this Merge $_{Meta}$ model that distinguish it from other models of meta-rights. Some of these characteristics are defined in literature (Barka & Sandhu, 2000b; L. Zhang et al., 2003; X. Zhang et al., 2003) for role to role membership transfer in RBAC (Ferraiolo & Kuhn, 1992; Ferraiolo et al., 2001; Sandhu et al., 1996). However, the values of these parameters are different in this research from the values proposed in the literature due to the nature of online social interactions and the ownership domain.

There are two basic axioms for this model, first, the model presented in this research is domain based as the meta-rights are associated with objects, and second, that no one can merge a meta-right associated with an object which they do not own. So, it is not possible for *Bob* to give *Alice* rights over another thread, where he does not have any meta-right.

[119] Also known as sub-forums.

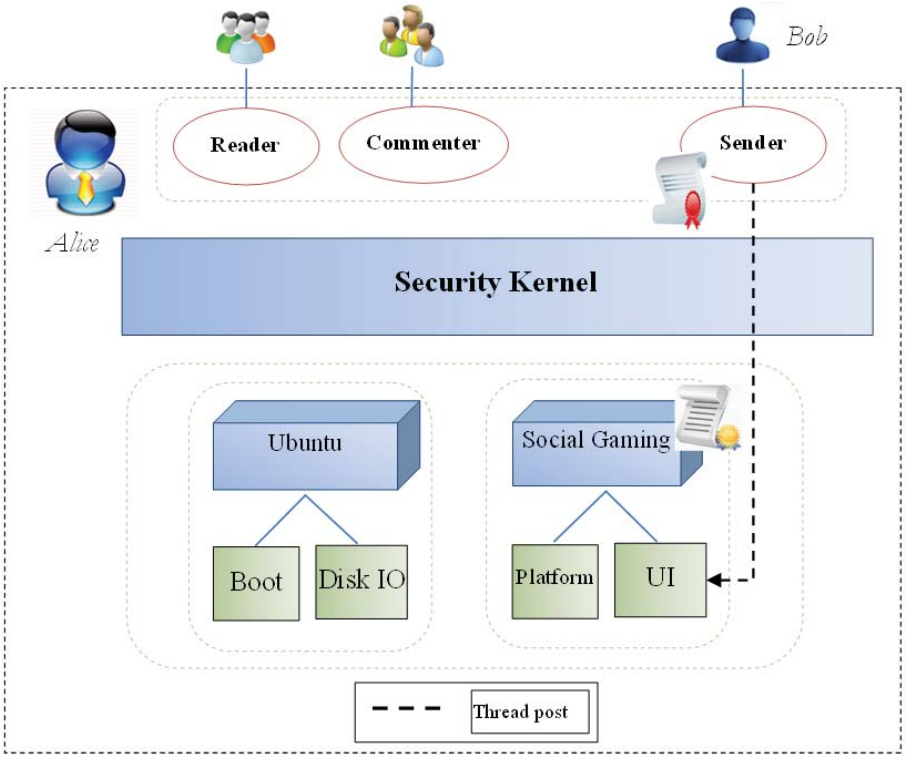

Figure 6.3: Meta-rights merge scenario depicting an Internet forum

6.3.1.1 Consent

Allocating the meta-right over a thread to *Alice* requires *Bob* to initiate the process as the system should enhance the owner trust. Also, it is necessary to take *Alice's* consent for every thread in her forum as she is responsible for its propagation and handling of its rights. Also, she must be aware of the object due to reputation management. If this condition is relaxed, one can share indecent threads within your forum without your consent.

The above implies that meta-rights are associated with one's repute[120]. So, a meta-right can only be merged with the consent of the owner and the beneficiary. It gives the following design principle:

P.6.3.1: Meta-rights merge requires the consent of the owner and the beneficiary.

[120] As discussed earlier in section 6.2.1.1.

6.3.1.2 Totality

Totality deals with allocating the proper/improper subset of rights over proper/improper subset of authorized resources. In the above example, it is up to *Bob* that how many meta-rights he wants to merge with *Alice* from his set of meta-rights. For example, only display meta-right for thread propagation can be merged so *Alice* cannot edit the thread nor can give the right to edit it to someone else, in which case the merge is partial or all the meta-rights over the thread can be merged where it is total. Partial merging of meta-rights is useful in various scenarios like only allowing the article publishers to decide about the display of articles but not let them edit it or display with their own name. Also multiple-partial merges can be used to achieve total merge of meta-rights using

$$Mrg_m(MR_{all}) = Mrg_m(MR_1) \wedge Mrg_m(MR_2) \wedge \ldots \wedge Mrg_m(MR_n) \ldots [eq.6.24]$$

Generalizing the above implies that there are two design possibilities for the Merge $_{Meta}$ model, that is, only complete meta-rights merge is supported so all the meta-rights or none can be merged with the beneficiary, or partial merge is supported so subset of all the meta-rights held by the owner can be merged. This model supports partial meta-rights merge as total can be achieved using partial and also there are scenarios where the original owner only wants to merge some of the meta-rights with the beneficiary, as display meta-rights in the above example. This gives the operational principle:

P.6.3.2: The owner can merge proper/improper subset of their rights.

6.3.1.3 Cardinality

Cardinality deals with the number of beneficiaries in a particular rights allocation operation. In the above example, *Bob* can merge the meta-rights over his thread with multiple actors say *Alice* and *Carl* at the same time[121]. Now if both of them agree to propagate it to the forum community only then the thread will be displayed, which gives

$$Mrg_m \rightarrow (MR_i)\{(Alice\&\ Bob\&Carl)|Bob,Carl \in VU \wedge Alice \in SH\ \forall VU, SH \in NS_i\} \ldots [eq.6.25]$$

[121] This supports the case when there are multiple moderators of the same forum. The moderators can't use the Share $_{Meta}$ model as it will exdusively give them meta-rights.

The propagation of information is only allowed if all the beneficiaries are agreed. This is also the case with public telephone directories where the telephone company and the telephone line owner both agree to display their telephone number on the list[122].

Generalizing the above case gives that a single meta-right can be merged with among multiple beneficiaries. This will not affect the object state but with higher number of beneficiaries, the agreement among all of them and thus the rights management becomes more difficult. It gives the design principle:

P.6.3.3: It is possible to merge a meta-right to more than one beneficiary.

6.3.1.4 Monotonicity

Monotonicity refers to whether the previous owner can exercise the right after the allocation. In the running example, if *Bob* merges meta-rights over his thread with *Alice*, both of them jointly manage the display of the thread. The case where the initiating party cannot exercise the right is covered in the Replace $_{Meta}$ model and the case when both of them can exercise at the same time is covered in the Share $_{Meta}$ model, so this Merge $_{Meta}$ model covers the cases when both can exercise it with the consent of each other. It merges the meta-rights of the entire actor set and gives

$$Mrg_m \rightarrow MR(\{Alice\ \&\ Bob\}, O, opr) \ldots [eq.\ 6.26]$$

Merge $_{Meta}$ model creates two roles with each merged right associated with the object, that is, the Primary Owner *(PO)* and the Joint Owner *(JO)*. It means that given x meta-rights associated with an object, the Merge $_{Meta}$ model can create up to $2x$ roles, which is also convenient for assigning rights to local roles. For actor centric implementations, the rights associated with them can be considered as

$$Mrg_m(JO) \rightarrow \begin{matrix} UR(\{JO\}, O, \phi) \\ MR(1/_N\{JO\}, O, opr) \end{matrix} \ldots [eq.\ 6.27]$$

and

[122] The telephone directory.

$$Mrg_m(PO) \rightarrow \begin{matrix} UR(\{PO\}, O, All) \\ MR(1/_N\{PO\}, O, opr) \end{matrix} \ldots [eq.6.28]$$

where N is the total number of beneficiaries for the particular meta-right. It can be generalized that if a meta-right of two actors is merged then both can exercise it with joint consent. This gives the division of authority and can be used to support separation of duties as one cannot complete the process without involving the other meta-right holder. This gives the design principle:

P.6.3.4: Joint consent of primary and joint owner is required to exercise the merged meta-right.

6.3.1.5 Depth

Depth deals with allowing the beneficiary to further pass on the allocated right. In the proposed model, as the joint owner cannot exercise the meta-rights alone, they cannot pass it to others but with the consent of all the other primary/joint owners. Meta-rights merge addressed in this model is single step which implies that the joint owner cannot further pass the right to another actor. This condition maintains the owner's authority and covers the options of sharing of information in moderated settings.

In the running scenario, *Alice* cannot further propagate the meta-right to *Carl*. If she initiates this, the system would not add *Carl* to the active actor set of the meta-right unless it is approved by *Bob*. This condition is necessary for system trust and gives the design principle:

P.6.3.5: Merging meta-rights does not allow the joint owner to further merge it.

6.3.2 Merge $_{\text{Meta}}$ process

The Merge $_{\text{Meta}}$ model maps various components of the SAC model to each other and defines the merge operation using the following set of rules:

a) The VU_j belongs to the LR_i in the NS_i

$$VU_j \rightarrow LR_i, \ \forall \ LR_i \in NS_i \ldots [eq.6.29]$$

b) The requested object O is classified into the same OC_τ.

$$O \ \rightarrow \ OC_\tau, \forall \ OC_\tau \in NS_i \ldots [eq.6.30]$$

c) The assignment of JO_i role to VU_j and PO_i to SH_i.

$$\begin{matrix} SH_i \rightarrow PO_i \\ VU_j \rightarrow JO_i \end{matrix}, \forall\ PO_i, JO_i \in LR_i \ldots [eq.6.31]$$

d) The merging of meta-rights with the JO_i, and restricting PO_i and JO_i to act jointly.

$$Mrg_u \rightarrow \begin{matrix} 1/2\ MR \rightarrow PO_i \\ 1/2\ MR \rightarrow JO_i \end{matrix} \forall\ MR \rightarrow O \ldots [eq.6.32]$$

Given a *NS i*, a merge request is only granted if the virtual user VU_j acquires a local role LR_i in namespace NS_i. Also the object *O* is mapped in object class *OC* with the privacy label τ. Further, VU_j is assigned to the joint owner role *JO*, which has the meta-rights over the object *O*.

6.3.3 Definition

Merge $_{Meta}$ is an operation that adds an actor to the actor set for a meta-right and merge the meta-right of the entire actor set. It can be defined as follows:

NS, SH, VU, Opr, LR, JO, PO, MR and *O* are sets of Namespaces, Stakeholders, Virtual Users, Operations, Local Roles, Joint Owner, Primary Owner, Meta-Rights and Objects respectively.

- A one-to-many stakeholder to Primary Owner role assignment relation is given by

$$SH\_PO \subseteq SH \times PO$$

- $assigned\_PO: (sh: SH) \rightarrow 2^{PO}$ is a function derived from *SH_PO*, for mapping of Primary Owner role onto stakeholder, which can be ϕ (empty) or the SH based on the current state of the object.

$$assigned\_PO\ (sh) = \{sh \in SH, po \in PO | (sh, po) \in SH\_PO, \forall\ SH, PO \in NS_i\}$$

- A many-to-many virtual user to Joint Owner role assignment relation is given by

$$VU\_JO \subseteq VU \times JO$$

- $assigned\_JO: (jo: JO) \rightarrow 2^{VU}$ is a function derived from *VU_JO*, for mapping of *JO* role over object to a set of virtual users.

$$assigned\_JO\ (jo) = \{vu \in VU, jo \in JO | (vu, jo) \in VU\_JO, \forall\ VU, JO \in NS_i\}$$

- $AC_m = 2^{(opr \times UR)}$ is the set of attestation certificates for meta-rights.
- A many-to-many *PO* role to attestation certificate assignment relation is given by

$$PO\_AC \subseteq PO \times AC_m$$

- $PO\_rights: (po: PO) \rightarrow 2^{AC_m}$ is a function derived from PO_AC_m for mapping

each Primary Owner role to a set of attestation certificates.

$$PO_{rights}(PO) = \{ac \in AC_m, po \in PO | (po, ac) \in PO\_AC_m, \forall\ PO, AC_m \in NS_i\}$$

- A many-to-many JO role to attestation certificate assignment relation is given by

$$JO\_AC_m \subseteq JO \times AC_m$$

- $JO\_rights: (jo: JO) \rightarrow 2^{AC_m}$ is a function derived from JO_MR, for mapping each Joint Owner role to a set of attestation certificates.

$$JO_{rights}(JO) = \{jo \in JO, ac \in AC_m | (jo, ac) \in JO_{AC_m}, \forall\ JO, AC_m \in NS_i\}$$

As all the meta-rights over an object can be merged with one or more joint owners, it gives the concept of active users over an object state, and can be calculated by

$$Active\ Users = PO \cup JO$$

Access Grant

A virtual user *vu* can perform an operation *opr* over an object *O*, under the Merge $_{Meta}$ model only if there exist a mapping of vu_j for the *JO* role over the object, and the *JO* role is authorized for the requested ac, so making vu_j a member of active user set, under the following relation:

$$vu : VU, AU: Active\ Users, ac: AC_m\ \forall\ VU, AU, AC_m \in NS_i$$

$$vu_j \in assigned\_JO\ (jo) \wedge ac \in JO\_rights(JO) \Longrightarrow vu_j \in AU$$

but an operation can only be performed with the consent of *PO*.

6.3.4 Rights analysis

This section describes the modification of rights for various roles of parent, offspring and general public associated with the merged object. Consider the thread moderated forum example, after submitting the thread to the group moderator, the thread is not moved from one topic/group to another, but remain in the same space – subgroup, so the rights of its parent, offspring and general public role remain the same. The rights of joint owner– the moderator are increased from none to a subset.

Below is table 6.6 for different rights of different roles associated with object before and after merge.

	Parent	PO	JO	Offspring	G. Public
Before Mrg $_{Meta}$	UR(V,D)	UR(V,D,E) MR(V,D,E)	D/C	UR(V)	D/C
After Mrg $_{Meta}$	UR(V,D)	UR(V,D,E) ½ MR(V,D,E)	UR(V) ½ MR(V,D,E)	UR(V)	D/C
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively PO represents primary owner, while JO represents joint owner D/C represents don't care and depends on beneficiary previous role in the space					

Table 6.6: Rights for different roles associated with object before and after the Merge $_{Meta}$ model

Below is table 6.7 for the roles of primary owner and joint owner, and their rights over parent, current, and child objects.

	PO			JO		
	Parent	**Mrg. Obj**	**Child**	**Parent**	**Mrg. Obj**	**Child**
Before Mrg $_{Meta}$	UR(V)	UR(V,D,E) MR(V,D,E)	UR(V,D)	D/C	D/C	D/C
After Mrg $_{Meta}$	UR(V)	UR(V,D,E) ½ MR(V,D,E)	½ UR(V,D)	UR(V)	UR(V) ½ MR(V,D,E)	½ UR(V,D)
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively PO represents primary owner, while JO represents joint owner D/C represents don't care and depends on beneficiary previous role in the space						

Table 6.7: Primary and Joint Owner rights over different objects before and after the Merge $_{Meta}$ model

A complete analysis of rights for different roles based on table 6.6 and 6.7 for the Merge $_{Meta}$ model is as follows:

a) After Merge $_{Meta}$, the primary owner rights over the object are reduced from having all meta-rights as single authority to joint authority.
b) The rights of primary owner over the parent space remain the same, that is, they can enter the parent space.
c) The rights of primary owner over child objects are also merged with the joint owner.

d) The rights of joint owner over the object are increased as they can now exercise some of the meta-rights[123] with the consent of the primary owner.
e) The joint owner also gets the view right over the parent spaces.
f) The joint owner gets view and joint delete right over the child objects.
g) Merging of meta-rights does not change the space of the object, so parent's rights over the object remain the same, that is, they can still view and delete the object.
h) Offspring role is associated with the object, so its rights after merging the rights of parent object are not changed, that is, they can view/enter the parent space.
i) General public role, its active users and their rights associated with the forum and thread remains the same, which depends on the parent and object configuration.

6.3.5 Design principles

The above discussion gives the following design principles for the Merge $_{Meta}$ model.

a) The model addresses domain based meta-rights merge where rights are associated with objects.
b) The owner can only merge the meta-rights over their own object.
c) Merge $_{Meta}$ gives only meta-rights[124], but use-rights remains with the primary owner.
d) Meta-rights merge requires the consent of both the primary and joint owner.
e) The owner can merge some/all of his meta-rights with the joint owner.
f) The same meta-right can be merged with multiple actors at the same time.
g) The primary owner can exercise the merged meta-right with the joint consent of the joint owner.
h) The joint owner cannot further pass the right as it would require joint consent of the primary owner.
i) After Merge $_{Meta}$, the rights of primary and joint owners are merged, so they can exercise it with the joint consent of each other. However, the rights of parent, offspring and general public role remain the same.

[123] Depending on the merged subset of rights.

[124] For example in case of public bulletin board, the administrator only has the right to display the message or not but cannot modify it.

6.3.6 Revocation

As the meta-right is merged with the joint-owner, it can only be revoked by their joint consent. In the moderated forum scenario, *Bob* can only takes back the meta-right from *Alice* if she is agreed to do it, which gives the design principle:

P.6.3.6: The primary owner can revoke the merged meta-rights with the consent of joint owner.

The model only supports half way self-revocation where both the primary owner and the joint owner must agree to revoke the right. Rule based revocation is not needed as the primary owner needs to confirm every action taken by the joint owner, which not allows him to violate any contract. Time based revocation is also ignored as the model is used in cases where online contents are mostly published and rarely changed. So, the model allows the primary owner to revoke the meta-right at will, and is allowed only if the joint owner agrees.

6.3.6.1 Self revocation

The primary owner requests the joint owner to revoke the right based on inappropriate use, difficulty in coordination or their will. In the above scenario, *Bob* can request *Alice* to remove the thread from display at any time at will, or delete it altogether. Allowing self-revocation gives the ease to the owner that a meta-right can be taken back if there is a deadlock between the joint owners.

6.3.6.2 The Merge $_{Meta}$ revoke process

The revocation removes the meta-rights from the joint owner *JO* and gives the full rights back to the Primary Owner, if no other joint owner is left for the particular meta-right over the object.

$$\begin{matrix} \neg MR \rightarrow JO_i \\ MR \rightarrow PO_i \end{matrix} \forall\, MR \rightarrow O \ldots [eq.6.33]$$

It also takes back the JO_i role from VU_j and PO_i role from SH_i.

$$\begin{matrix} \neg\, JO_i \rightarrow VU_j \\ \neg\, PO_i \rightarrow SH_i \end{matrix} \forall\, JO_i, PO_i \in LR \wedge VU_i, SH_i \in NS_i \ldots [eq.6.34]$$

6.3.7 Summary of Merge $_{Meta}$

To summarize, the Merge $_{Meta}$ model allow the owner of an object to merge the meta-rights of all the actors and their joint consent is needed to exercise that right. The characteristics are given in section 6.3.1, which distinguish the model from other models of use-rights allocation. Section 6.3.2 explains the allocation process, followed by the logical definition of the model in section 6.3.3. Section 6.3.4 illustrates the rights analysis, the design principles are mentioned in section 6.3.5 while the revocation is discussed in section 6.3.6.

6.4 Chapter summary

This chapter has discussed the meta-rights allocation in online social interactions by outlining the details of Replace $_{Meta}$, Share $_{Meta}$, Merge $_{Meta}$ models. It has also outlined the revoke process in each case. The Replace $_{Meta}$ model was demonstrated with the copyright example of research papers, Share $_{Meta}$ model was explained using video sharing on Facebook wall, while Merge $_{Meta}$ was explained with threads in Internet discussion forum example.

Chapter 7

Analysis

This chapter analyzes the use-rights and meta-rights models presented in chapter 5 and 6. It contains some important general observations about the behavior of the models, their precedence over each other and the effect of that precedence. It starts with the possible permutations of the basic models and their precedence over each other. It then outlines the similarities of the use-rights and meta-rights models in terms of their output, characteristics, and rights of various roles associated with the object. If further generalizes the notion of rights triplet to incorporate the semantics of both use-rights and meta-rights.

7.1 Model permutations and their precedence

In the previous chapters, various use-rights and meta-rights models for online social interactions are outlined. However, they only cover the basic scenarios and a complex policy may require some complex allocation model – built on a combination of these basic models. Conversely, as any combination of two, three, or four models can result in a new complex model[125], it may not be possible to incorporate all the possible complex models in one concrete piece of work. However, the outlined basic models can be used in combinations to

[125] With new scenarios, distinct characteristics values and different rights of the entire role set.

provide any desired complex allocation. This section outlines all the possible permutations of the use-rights and meta-rights models presented in chapter 5 and 6.

To generate all the possible permutations[126], the models are united keeping the upper limit constant. The permutations used are ${}^{n}P_{r}$ with n as 4[127] and r as 1, 2, 3 and 4 for separate permutation upper limits[128]. The resultant permutations are ${}^{4}P_{1}$, ${}^{4}P_{2}$, ${}^{4}P_{3}$, and ${}^{4}P_{4}$, which give 4, 12, 24 and 24. This gives all the permutations using a single model – 4 possible permutations, any two models – 12 possible permutations, any three models – 24 possible permutations, and all the four models – 24 possible permutations. This results in total 64 possible permutations for use rights and the same number of permutations for meta-rights. Given below are these permuted complex models and their possible outcomes followed by some interesting observations.

7.1.1 Use-Rights permutations

In chapter 5, this dissertation has outlined four models of use-rights, that is, Replace $_{Use}$, Share $_{Use}$, Merge $_{Use}$ and Revoke $_{Use}$. Revoke $_{Use}$ has different characteristics for each of the three models but for generalization, here it is considered as a single model – that revokes the already allocated right from a user. The rules for generating these permuted complex models for use-rights are taken from the basic models' definitions in chapter 5, and are as follows:

a) Replace *(Rep)* – Replaces the existing actor set with a new actor, provided as the function argument.
b) Share *(Shr)* – Adds an actor to the actor set and allocates them the same rights as of the previous actors, without changing their existing rights.
c) Merge *(Mrg)* – Merges the rights of the entire actor set.
d) Revoke *(Rev)* - Removes the actor provided as the function argument, from the use-right and adds the owner if the actor set is empty.

Table 7.1 present in appendix-A, shows the possible permutations in every possible order for use-rights based on the basic use-rights models presented in chapter 5. The table and its

[126] Permutations consider the order during arrangement, where (x, y) is different than (y, x).

[127] Four types of rights allocation models, that is, Replace $_{Use}$, Share $_{Use}$, Merge $_{Use}$ and Revoke $_{Use}$.

[128] Permutations under single upper limit is calculated by $\frac{n!}{n-r!}$.

analysis has two benefits: First, it provides the opportunity of having all the models jointly (complex models) for complex scenarios, which has enormous possible outcomes[129], by using only four basic models. Second, it provides the complete description of the outcomes that can possibly arise in an access control state after applying a use-rights model followed by any other use-rights model. Here *MO* stands for meta-owner – the owner of the object having all the rights, and *A, B* and *C* are any arbitrary actors in the system. Also actors as argument in a model function are allocated some rights, and target is the actor being affected by those allocations. Table 7.1 can be easily understood by figure 7.1, where model represents the sequence of use-rights model application, start state represents the state of rights triplet before allocation, target represents the actor that will affect by the allocation and end state represents the state of rights triplet after the particular use-rights allocation. The dotted arrows show that the previous end state is used as the start state for the next allocation. The figure has been drawn for the Replace $_{Use}$ model only by keeping the Share $_{Use}$ model at the second position in the permutation, where similar figures can be drawn for Share $_{Use}$, Merge $_{Use}$ and Revoke $_{Use}$ models.

7.1.1.1 Observations

Following are some of the observations drawn from table 7.1 and the precedence[130] of use-rights allocation models over each other:

a) If the Replace $_{Use}$ model is applied after any model, it removes the previous features and completely applies its own.
b) The Share $_{Use}$ model adds its features but also keeps the features of the previous applied models.
c) Applying the Share $_{Use}$ model on previously merged use-rights just adds another actor but the use-rights remain merged.
d) The Merge $_{Use}$ model adds its features but also maintains the features of the previous applied models.

[129] For example, (Replace $_{Use}$ →Merge $_{Use}$) has different output than (Replace $_{Use}$ →Share $_{Use}$), which are different from (Replace $_{Use}$ →Merge $_{Use}$ →Share $_{Use}$) and (Replace $_{Use}$ →Share $_{Use}$ →Merge $_{Use}$).

[130] The effectiveness of one allocation model over others.

e) Applying the Merge $_{Use}$ model on already shared use-rights does not add any actor but merges the use-rights among the existing actors.

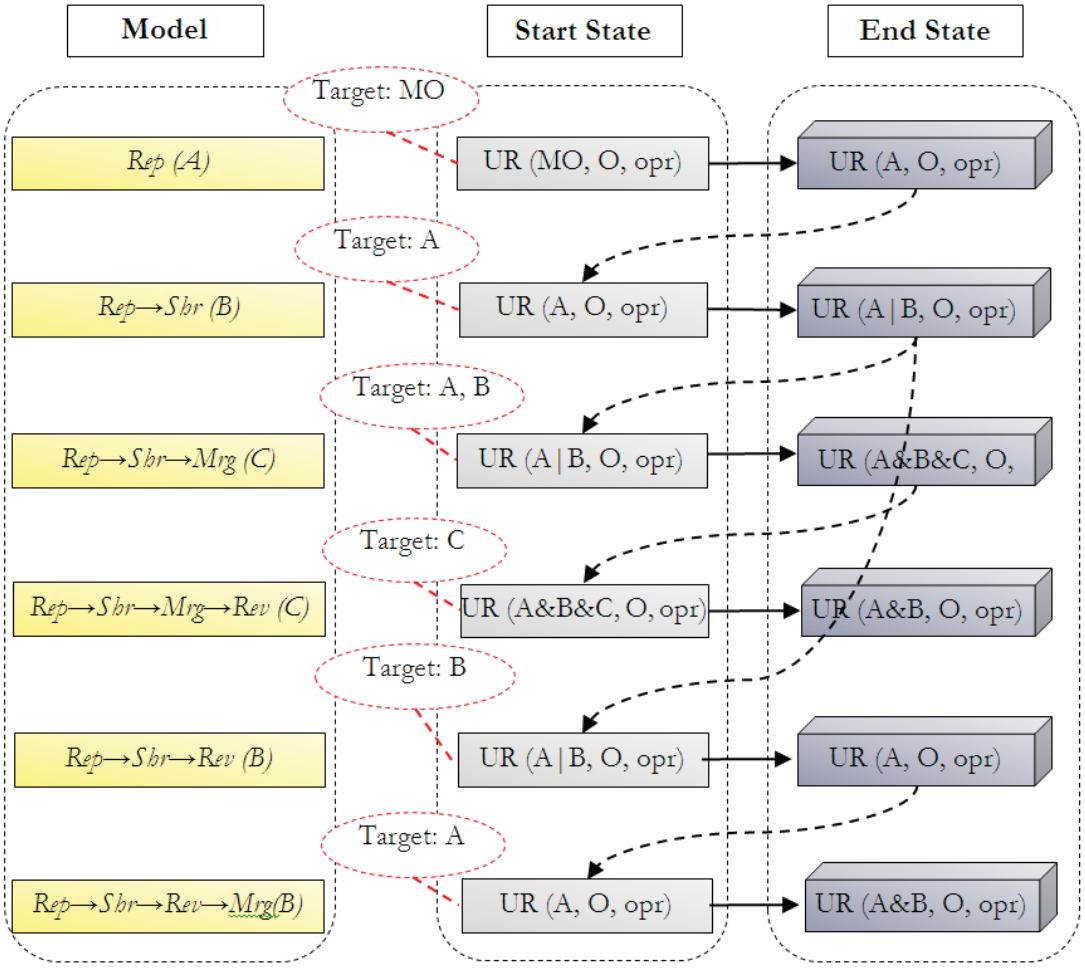

Figure 7.1: Visual description of formation of use-rights allocation permutation table

f) The Revoke $_{Use}$ model removes the features of the previous applied models and applies its own features.

g) The Revoke $_{Use}$ and Replace $_{Use}$ models are similar as they remove the previous features and apply their own, while the Share $_{Use}$ and Merge $_{Use}$ models are similar as they keep the features of the previous models and also add their own features.

The precedence of these models over each other is illustrated in table 7.2.

	Replace $_{Use}$	***Share*** $_{Use}$	***Merge*** $_{Use}$	***Revoke*** $_{Use}$
Rep	-	*Rep*	*Rep*	*Rep*
Shr	*Rep, Shr*	-	*Mrg, Shr*	*Rev, Shr*
Mrg	*Rep, Mrg*	*Shr, Mrg*	-	*Rev, Mrg*
Rev	*Rev*	*Rev*	*Rev*	-

Table 7.2: Precedence of various use-rights model applications over each other

7.1.2 Meta-Rights permutations

Chapter 6 of this dissertation has sketched four models of meta-rights, that is, Replace $_{Meta}$, Share$_{Meta}$, Merge $_{Meta}$ and Revoke $_{Meta}$. Revoke $_{Meta}$ has different characteristics for each of the three models but for generalization, here it is considered as a single model – that revokes the already allocated right from a user. The rules for generating these permuted complex models for meta-rights are taken from the basic models' definitions in chapter 6, and are same as the rules illustrated in section 7.1.1. Table 7.3 present in appendix-A, shows the possible permutations in every possible order for meta-rights based on the basic meta-rights models presented in chapter 6. The table can be easily understood by figure 7.1 drawn for use-rights permutations.

7.1.2.1 Observations

Following are some of the observations drawn from table 7.3 and the precedence of meta-rights allocation models over each other:

a) If the Replace $_{Meta}$ model is applied after any model, it removes the previous features and completely applies its own.
b) The Share $_{Meta}$ model adds its features but also maintains the features of the previous applied models.
c) Applying the Share $_{Meta}$ model on previously merged meta-rights just adds another actor but the meta-rights remain merged.
d) The Merge $_{Meta}$ model adds its features but also maintains the features of the previous applied models.
e) Applying Merge $_{Meta}$ model on already shared meta-rights does not add any actor but merges the meta-rights among the existing actors.

f) The Revoke $_{Meta}$ model removes the features of the previous applied models and applies its own features.

g) The Revoke $_{Meta}$ and Replace $_{Meta}$ models are similar as they remove the previous features and apply their own, while the Share $_{Meta}$ and Merge $_{Meta}$ models are similar as they keep the features of the previous models and also add their own features.

The precedence of these models over each other is illustrated in table 7.4.

	Replace $_{Meta}$	***Share*** $_{Meta}$	***Merge*** $_{Meta}$	***Revoke*** $_{Meta}$
Rep	-	*Rep*	*Rep*	*Rep*
Shr	*Rep, Shr*	-	*Mrg, Shr*	*Rev, Shr*
Mrg	*Rep, Mrg*	*Shr, Mrg*	-	*Rev, Mrg*
Rev	*Rev*	*Rev*	*Rev*	-

Table 7.4: Precedence of various meta-rights model applications over each other

7.2 Similarities between use-rights and meta-rights models

In previous chapters, this dissertation took the same approach as some of the previous studies that meta-rights are different from use-rights[131] (Barka, 2002; Barka & Sandhu, 2004, 2007; Crampton & Khambhammettu, 2008; Dewan & Shen, 1998; Indratmo & Vassileva, 2007; L. Zhang et al., 2001, 2003; Mattas et al., 2006; Munawer, 2000; Park, 2003; Sandhu, Bhamidipati & Munawer, 1999; Sandhu & Munawer, 1998a, 1998b, 1999b; X. Zhang et al., 2003). However, after analyzing the models' precedence over each other in table 7.2 and 7.4, along with the outcome of their applications in tables 7.1 and 7.3, this research recommends the generalization of the use-rights and meta-rights allocation models. This section now discusses the possibility that use-rights and meta-rights allocation models are not very different and their objective can be achieved using one generalized kind. Meta-rights allocation models deal with only special types of rights which are involved in the administration of rights but their other characteristics, and their behavior is quite similar to those of use-rights allocation models[132]. Following are some of the similarities between these use-rights and meta-rights models:

[131] Meta-rights deal with the administration of rights.

[132] This argument leads to the generalization of delegation and transfer models in traditional models.

7.2.1 Replace model

Replace model replaces the previous actor set with a new actor, whether it is applied on use-rights or meta-rights, and is done by replacing the current actor set with the single beneficiary in the rights triplet. The replace function can be represented by

$$Rep\,(A) \rightarrow R\,(\{D/C\}, O, opr) \Rightarrow R\,(\{A\}, O, opr) \ldots [eq.\,7.1]$$

where D/C stands for don't care and shows that whoever is in the rights triplet is replaced by the new actor.

Following is some analysis related to various properties of the Replace model, which give insights about the generalized model, predict its behavior, and show the stability of the access control state after applying that model. It also shows that if the system is in stable state as initial state, and one applies any combination of these allocations, the system still remains in the stable state.

7.2.1.1 Similarity analysis

The similarity between the Replace $_{Use}$ and Replace $_{Meta}$ models can be analyzed by the following three arguments: a) the similarity between their output, b) the similarity between their characteristics, and c) the similarity between their effects on rights of different roles. These arguments are given as follows:

Argument 1: Both the Replace models have similar output if applied on same right.

The outputs of both the Replace models are similar in nature as can be seen from table 7.1 and 7.3, and summarized in table 7.5, where applying the Replace $_{Use}$ model and the Replace $_{Meta}$ model results in similar output.

Use-Rights and Meta-Rights			
Allocation Sequence	**Start State**	**Target**	**End State**
Replace $_{Use}$ (A)	*UR({MO}, O, opr)*	*MO*	*UR({A}, O, opr)*
Replace $_{Meta}$ (A)	*MR({MO}, O, opr)*	*MO*	*MR({A}, O, opr)*

Table 7.5: Output summary of Replace $_{Use}$ and Replace $_{Meta}$ models

$$Rep_{Use} \rightarrow UR(\{A\}, O, opr) \ldots [eq.7.2(a)]$$

$$Rep_{Meta} \rightarrow MR(\{A\}, O, opr) \ldots [eq.7.2(b)]$$

$$UR(\{A\}, O, opr) \approx MR(\{A\}, O, opr) \Rightarrow Rep_{Use} \approx Rep_{Meta} \ldots [eq.7.2(c)]$$

Argument 2: Both the Replace models have similar characteristics.

To highlight the similarities between the two types, table 7.6 depicts various characteristics of Replace $_{Use}$ and Replace $_{Meta}$ models.

	Consent	Totality	Cardinality	Monotonicity	Depth	Revocation
Replace $_{Use}$	Both	Partial	One	Mut. Exd.	Single pass	S/T/V
Replace $_{Meta}$	Both	Partial	One	Mut. Exd.	Chain	No

Table 7.6: Comparison of various characteristics of Replace model

From the above table, some insights are drawn and are as follows:

a) Consent of the owner and the beneficiary is required for both Replace models, which means that Replace model can only be used with owner and beneficiary consent.
b) Both the Replace models operate on partial subset of rights.
c) Both the Replace models have singular cardinality as they replace the actor with single beneficiary.
d) The owner can no longer exercise the replaced right, so both the models are mutually exclusive.
e) As meta-rights give the authority to the beneficiary to further allocate the right while use-rights does not allow it, depth is associated with the nature of meta-rights and does not affect the allocation process.
f) Another difference between the two models is revocation, which can be seen from tables 7.1 and 7.3 as well. However, revocation is also directly concerned with the nature of meta-rights where if the owner remains with no meta-right, they cannot exercise it (revoke it). Also as the beneficiary becomes the new owner, they can revoke the rights of other actors.

There are two differences between these Replace models. If it is generalized for both types of rights, then it should take these characteristics – depth and holder of the meta-right, as parameters.

Argument 3: Both the Replace models have similar effect on rights for different roles.

To highlights the similarities of rights allocations between the Replace $_{Use}$ and Replace $_{Meta}$ models, table 7.7 depicts the rights for every role associated with the object for both the Replace models.

		Parent	Owner	Beneficiary	Offspring	G. Public
Before		UR (V,D)	UR (V,D,E) MR(V,D,E)	D/C	UR (V)	D/C
After	**Rep** $_{Use}$	UR (V,D)	UR (V) MR(V,D,E)	UR (V,D,E)	UR (V)	D/C
	Rep $_{Meta}$	NR	NR	UR (V,D,E) MR(V,D,E)	UR (V)	D/C
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively D/C represents don't care NR represents No Right						

Table 7.7: Rights of different roles associated with the objects before and after Replace model

Further, table 7.8 depicts the rights of owner and beneficiary over different objects for both types of Replace model.

Replace		**Owner**			**Beneficiary**		
		Parent	**Object**	**Child**	**Parent**	**Object**	**Child**
Before		UR(V)	UR(V,D,E) MR(V,D,E)	UR(V,D)	D/C	D/C	D/C
After	**Use**	UR(V)	UR(V) MR(V,D,E)	UR(V)	UR(V)	UR(V,D,E)	UR(V,D)
	Meta	NR	NR	NR	UR(V)	UR(V,D,E) MR(V,D,E)	UR(V,D)
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively D/C represents don't care NR represents No Right							

Table 7.8: Owner and beneficiary rights over different objects before and after Replace model

From the above tables, some insights are drawn and are as follows:

a) The owner completely loses the replaced right over the object in both models, whether it is edit, delete or meta-right.
b) If owner has some right over their object, they remain with view right over the parent space. However if they lose all the rights over the object then they also lose all the rights over the parent space. The owner's rights over the parent space are similar for both models, keeping this assumption in place.
c) The owner loses control over child object in both models, however if they have some right over the object, then they can view the child object.
d) The beneficiary gets the replaced right over the object from the owner in both models.
e) The beneficiary gets the view right over the parent spaces in both models.
f) The beneficiary gets the same rights over the child object in both models.
g) The rights of parent role over the object remain the same for both replace models; however the actors in the old owner's parent role are replaced with the actors in the new owner's parent role.
h) The rights of offspring role remain the same after both the Replace models.
i) The actors in the old general public role are replaced with the actors in new general public role; however, their rights may change depending upon the configuration of the new space.

Based on the above, the definition of rights can be generalized to make it consistent for use rights as well as meta-rights. As the major difference is of the allowed depth of the model along with the meta-rights authority, it is good to include both of them in the rights equation. It will change the rights equation by

$$Right\ (Actor, Object, Operation, Depth, Owner) \ldots [eq.\ 7.3]$$

where the first three variables are the same as traditional rights equation. The newly introduced depth relates to the capability of the beneficiary to further allocate a right or not. If it is 0, the actor cannot further allocate the right, while if it is 1, the actor is allowed to further pass the

right on. The last variable is the owner of the object[133] and its modification is associated with the allocation of meta-rights over the object.

This generalized equation can be used for the Replace model and can be represented as

$$Right\,(Dge, O, view, 0, Dgr) \ldots [eq.\,7.4]$$

where 0 mentions that it is the replace of use-right and the delegator is still the owner of the object.

7.2.1.2 Precedence analysis

Replace model has higher precedence over other rights allocation models. A rights allocation model has higher precedence if two different allocations sequences *x* and *y* ending at the same model achieve the same final state with respect to rights as well as characteristics.

The rights output of Replace model does not depend on the order of applications of previous rights allocation models. Thus it has high precedence as the right's state of two different allocation sequences ending at Replace are same, which can be seen in table 7.1 and 7.3 and summarized in table 7.9.

Allocation Sequence	Start State	Target	End State
Shr → Rep (B)	*UR({MO|A}, O, opr)*	*MO, A*	*UR({B}, O, opr)*
Mrg → Rep (B)	*UR({MO&A}, O, opr)*	*MO, A*	*UR({B}, O, opr)*
Rev → Rep (B)	*UR({MO}, O, opr)*	*MO*	*UR({B}, O, opr)*

Table 7.9: Output Summary of the Replace model

Also, as highlighted in table 7.2 and 7.4, and summarized in table 7.10, if Replace model is applied after any model, it overrides the characteristics of that model with its own.

	Share	***Merge***	***Revoke***
Rep	*Rep*	*Rep*	*Rep*

Table 7.10: Precedence of Replace model over other rights allocation models

[133] For ownership domain in particular.

7.2.1.3 Order analysis

The change in order of multiple applications of the Replace model affects the final outcome. The order of multiple applications of Replace model does matter and the output changes depending on the order of input. It is visible from tables 7.1 and 7.3, and can also be seen from the following two equations, where

$$Rep\ (A) \rightarrow Rep\ (B) \rightarrow Rep\ (C) = R(\{C\}, O, opr) \ldots [eq.\ 7.5]$$

results in different final stage as

$$Rep\ (C) \rightarrow Rep\ (B) \rightarrow Rep\ (A) = R(\{A\}, O, opr) \ldots [eq.\ 7.6]$$

7.2.2 Share model

Share is a function that adds a new member to the actor set without changing the previous actors, whether it is applied on use-rights or meta-right, and is done by adding the beneficiary to the actor set for the shared right. The Share function can be described as

$$Shr\ (B) \rightarrow R\ (\{A\}, O, opr) \Rightarrow R\ (\{A, B\}, O, opr) \ldots [eq.7.7]$$

where A is added to the actor set for the specific right without changing the previous actor B.

Following is some analysis related to various properties of the Share model, which give insights about the generalized model, predict its behavior, and show the stability of the access control state after applying that model. It also shows that if the system is in stable state as initial state, and one applies any combination of these allocations, the system still remains in the stable state.

7.2.2.1 Similarity analysis

The similarity between the Share $_{Use}$ and Share $_{Meta}$ models can be analyzed by the following three arguments: a) the similarity between their output, b) the similarity between their characteristics, and c) the similarity between their effects on rights of different roles. These arguments are given as follows:

Argument 1: Both the Share models have similar output if applied on same right.

The outputs of both the Share models are similar in nature as can be seen from table 7.1 and 7.3, and summarized in table 7.11, where applying the Share $_{Use}$ model and Share $_{Meta}$ model results in similar output.

Use-Rights and Meta-Rights			
Allocation Sequence	**Start State**	**Target**	**End State**
Share $_{Use}$ (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO|A}, O, opr)*
Share $_{Meta}$ (A)	*MR({MO}, O, opr)*	*MO*	*MR({MO|A}, O, opr)*

Table 7.11: Output summary of Share $_{Use}$ and Share $_{Meta}$ models

$$Shr_{Use} \rightarrow UR(\{MO|A\}, O, opr) \ldots [eq.7.8\ (a)]$$

$$Shr_{Meta} \rightarrow MR(\{MO|A\}, O, opr) \ldots [eq.7.8\ (b)]$$

$$UR(\{MO|A\}, O, opr) \approx MR(\{MO|A\}, O, opr) \Rightarrow Shr_{Use} \approx Shr_{Meta} \ldots [eq.7.8\ (c)]$$

Argument 2: Both the Share models have similar characteristics.

To highlight the similarities between the two types, table 7.12 depicts various characteristics of Share $_{Use}$ and Share $_{Meta}$ models

	Consent	**Totality**	**Cardinality**	**Monotonicity**	**Depth**	**Revocation**
Share $_{Use}$	Owner	Partial	Many	Mut. Ind.	Single pass	S/T/V
Share $_{Meta}$	Both	Partial	Many	Mut. Ind.	Chain	Self (Both)

Table 7.12: Comparison of various characteristics of Share model

From the above table, some insights are drawn and are as follows:

a) Consent of both the owner and the beneficiary is required for meta-rights model while it is not necessary for use-right model. The use-rights model is often used for giving away rights and making things publically available, so it is efficient and scalable not to take the beneficiary consent for use-rights sharing.
b) Both the Share models operate on partial subset of rights.
c) Both the Share models have multiple-cardinality as they share the rights with multiple beneficiaries.

d) Both the Share models are mutually inclusive, that is, owner and the beneficiary both can exercise the shared right.

e) As meta-rights give the authority to the beneficiary to further allocate the right while use-rights does not allow it, depth is associated with the nature of meta-rights and does not affect the allocation process.

f) Both the Share models allow the owner to revoke of rights from the beneficiary.

The only prominent difference between the two Sharing models is the depth of rights allocation. If it is generalized for both types of rights, then it should take depth as a parameter.

Argument 3: Both the Share models have similar effect on rights for different roles.

To highlight the similarities of rights allocations between Share $_{Use}$ and Share $_{Meta}$ models, table 7.13 depicts the rights for every role associated with the object for both the Share models.

		Parent	Owner	Beneficiary	Offspring	G. Public
Before		UR (V,D)	UR (V,D,E) MR(V,D,E)	D/C	UR (V)	D/C
After	**Shr $_{Use}$**	UR (V,D)	UR (V,D,E) MR(V,D,E)	UR (V,D,E)	UR (V)	D/C
	Shr $_{Meta}$	UR (V,D)	UR (V,D,E) MR(V,D,E)	UR (V,D,E) MR(V,D,E)	UR (V)	D/C

V, D and E represents view, delete and edit respectively
UR and MR represents use-rights and meta-rights respectively
D/C represents don't care
NR represents No Right

Table 7.13: Rights of different roles associated with the objects before and after Share model

Further, table 7.14 depicts the rights of owner and beneficiary over different objects for both the Share models.

Share		Owner			Beneficiary		
		Parent	Object	Child	Parent	Object	Child
Before		UR(V)	UR(V,D,E) MR(V,D,E)	UR(V,D)	D/C	D/C	D/C
After	Use	UR(V)	UR(V,D,E) MR(V,D,E)	UR(V,D)	UR(V)	UR(V,D,E)	UR(V,D)
	Meta	UR (V)	UR(V,D,E) MR(V,D,E)	UR (V,D)	UR (V)	UR(V,D,E) MR(V,D,E)	UR (V,D)
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively D/C represents don't care							

Table 7.14: Owner and beneficiary rights over different objects before and after Share model

From the above tables, some insights are drawn and are as follows:

a) In both models, the owner's rights over the object, parent space, and the child object remain the same with no change in their previous state.
b) The beneficiary gets the shared right over the object in both models[134].
c) The beneficiary gets the view right over the parent space in both models.
d) The beneficiary gets the view and delete rights over the child object in both models.
e) The rights of parent role remain the same after both the Share models.
f) The rights of offspring role do not change after both the Share models.
g) The rights of general public role remain the same after both the Share models.

Based on the above, the definition of rights can be generalized to make it consistent for use rights as well as meta-rights. As the major difference is of the allowed depth of the model along with the meta-rights authority, it is good to include both of them in the rights equation. It will change the rights equation by

$$Right\ (Actor, Object, operation, Depth, Owner) \ldots [eq.\ 7.9]$$

where the first three variables are the same as traditional rights equation. The newly introduced depth relates to the capability of the beneficiary to further allocate a right or not. If it is 0, the

[134] As sharing view right gives only view right and sharing edit right gives only edit right, and both rights are different but the model is the same, that is, Share $_{Use}$. The same applies in case of use-rights and meta-rights, where the rights are different but the models' procedure is the same.

actor cannot further allocate the right, while if it is 1, the actor is allowed to further pass the right on. The last variable is the owner of the object and its modification is associated with the allocation of meta-rights over the object.

This generalized equation can be used for the Share model and can be represented as

$$Right\ (\{Ben1, Ben2\}, O, view, 0, Owner) \ldots [eq.\ 7.10]$$

where 0 mentions that it is the share of use-right and the owner is still the actual authority over the object. Also the sharing of meta-rights with the secondary owner can be represented as

$$Right\ (Ben, O, view, 1, \{PO, SO\}) \ldots [eq.\ 7.11]$$

7.2.2.2 Precedence analysis

Share model has lower precedence over other rights allocation models. A rights allocation model has higher precedence if two different allocations sequences *x* and *y* ending at the same model achieve the same final state with respect to rights as well as characteristics.

The rights output of Share model does depend on the order of applications of previous rights allocation models. Thus it has low precedence as the rights state of two different allocation sequences ending at Share are different, which can be seen in table 7.1 and 7.3 and summarized in table 7.15.

Use-Rights			
Allocation Sequence	**Start State**	**Target**	**End State**
Rep → Shr (B)	*UR({A}, O, opr)*	*A*	*UR({A|B}, O, opr)*
Mrg → Shr (B)	*UR({MO&A}, O, opr)*	*MO,A*	*UR({MO&A&B}, O, opr)*
Rev → Shr (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO|A}, O, opr)*

Table 7.15: Output Summary of the Share model

Also, as highlighted in table 7.2 and 7.4, and summarized in table 7.16, if Share model is applied after any model, it adds its features but also keeps the features of the previous model.

	Replace	**Merge**	**Revoke**
Shr	Rep, Shr	Mrg, Shr	Rev, Shr

Table 7.16: Precedence of Share model applications over other rights allocation models

7.2.2.3 Order analysis

The change in order of multiple applications of the Share model does not affect the final outcome. The order of multiple applications of Share model does not matter and the output remains the same by changing the order of input. It is visible from tables 7.1 and 7.3, and can also be seen from the following two equations, where

$$Shr\,(A) \rightarrow Shr\,(B) \rightarrow Shr\,(C) = R(\{A|B|\,C\}, O, opr) \ldots [eq.\,7.12]$$

results in the same final stage as

$$Shr\,(C) \rightarrow Shr\,(B) \rightarrow Shr\,(A) = R(\{A\mid B\mid C\}, O, opr) \ldots [eq.\,7.13]$$

7.2.3 Merge model

Merge is a function that (adds a new actor to the actor set if not present and then) merges the rights of the complete actor set, whether it is applied on use-rights or meta-rights. The Merge function can be represented by

$$Mrg\,(A) \rightarrow R\,(\{B\}, O, opr) \Rightarrow R\left({}^{1}/_{2}\{A, B\}, O, opr\right) \ldots [eq.\,7.14]$$

where A is added to the actor set and the rights of A are merged with all the previous actors, B in this case.

Following is some analysis related to various properties of the Merge model, which give insights about the generalized model, predict its behavior, and show the stability of the access control state after applying that model. It also shows that if the system is in stable state as initial state, and one applies any combination of these allocations, the system still remains in the stable state.

7.2.3.1 Similarity analysis

The similarity between the Merge $_{Use}$ and Merge $_{Meta}$ models can be analyzed by the following three arguments: a) the similarity between their output, b) the similarity between their characteristics, and c) the similarity between their effects on rights of different roles. These arguments are given as follows:

Argument 1: Both the Merge models have similar output if applied on same right.

The outputs of both the Merge models are similar in nature as can be seen from table 7.1 and 7.3, and summarized in table 7.17, where applying Merge $_{Use}$ model and Merge $_{Meta}$ model results in similar output.

	Use-Rights and Meta-Rights		
Model	**Start State**	**Target**	**End State**
Merge $_{Use}$ (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO&A}, O, opr)*
Merge $_{Meta}$ (A)	*MR({MO}, O, opr)*	*MO*	*MR({MO&A}, O, opr)*

Table 7.17: Output summary of Merge $_{Use}$ and Merge $_{Meta}$ models

$$Mrg_{Use} \rightarrow UR(\{MO\&A\}, O, opr) \ldots [eq.7.15\ (a)]$$

$$Mrg_{Meta} \rightarrow MR(\{MO\&A\}, O, opr) \ldots [eq.7.15\ (b)]$$

$$UR(\{MO\&A\}, O, opr) \approx MR(\{MO\&A\}, O, opr) \Rightarrow Mrg_{Use} \approx Mrg_{Meta} \ldots [eq.7.15\ (c)]$$

Argument 2: Both the Merge models have similar characteristics.

To highlight the similarities between the two types, table 7.18 depicts various characteristics of Merge $_{Use}$ and Merge $_{Meta}$ models.

	Consent	**Totality**	**Cardinality**	**Monotonicity**	**Depth**	**Revocation**
Merge $_{Use}$	Both	Partial	Many	Joint Ind.	Single	S/T
Merge $_{Meta}$	Both	Partial	Many	Joint Ind.	Jointly Chain	Self (Joint)

Table 7.18: Comparison of various characteristics of Merge model

From the above table, some insights are drawn and are as follows:

a) Consent of the owner and the beneficiary is required for both Merge models.
b) Both the Merge models operate on partial subset of rights
c) Both the Merge models have multiple-cardinality as they merge a single right with multiple beneficiaries.
d) Both the Merge models are joint mutually inclusive, that is, owner and the beneficiary both can exercise the right with the consent of each other.

e) As meta-rights give the authority to the beneficiary to further allocate the right[135] while use-rights does not allow it, depth is associated with the nature of meta-rights and does not affect the allocation process.

f) Both the Merge models allow the owner to revoke of rights from the beneficiary but in meta-rights revocation, their consent is required as well.

The only prominent difference between the two Merge models is the depth of rights allocation. If it is generalized for both types of rights, then it should take depth as a parameter.

Argument 3: Both the Merge models have similar effect on rights for different roles.

To highlight the similarities of rights allocations between Merge $_{Use}$ and Merge $_{Meta}$ models, table 7.19 depicts the rights for every role associated with the object for both the Merge models.

		Parent	Owner	Ben	Offspring	G. Public
Before		UR(V,D)	UR (V,D,E) MR(V,D,E)	D/C	UR (V)	D/C
After	**Mrg** Use	UR(V,D)	½ UR(V,D,E) MR(V,D,E)	½ UR(V,D,E)	UR (V)	D/C
	Mrg Meta	UR(V,D)	UR (V,D,E) ½ MR(V,D,E)	UR (V) ½ MR(V,D,E)	UR (V)	D/C
V, D and E represents view, delete and edit respectively UR and MR represents use-rights and meta-rights respectively D/C represents don't care NR represents No Right						

Table 7.19: Rights of different roles associated with the objects before and after Merge model

Table 7.20 shows the rights of owner and beneficiary over different objects for both the Merge models.

[135] With the consent of the owner.

Merge		Owner			Beneficiary		
		Parent	Object	Child	Parent	Object	Child
Before		UR(V)	UR(V,D,E) MR(V,D,E)	UR(V,D)	D/C	D/C	D/C
After	Use	UR(V)	½UR(V,D,E) MR(V,D,E)	½ UR(V,D)	UR(V)	½ UR(V,D,E)	½ UR(V,D)
	Meta	UR(V)	UR(V,D,E) ½MR(V,D,E)	½ UR(V,D)	UR(V)	UR(V) ½ MR(V,D,E)	½ UR(V,D)

V, D and E represents view, delete and edit respectively

UR and MR represents use-rights and meta-rights respectively

D/C represents don't care

NR represents No Right

Table 7.20: Owner and beneficiary rights over different objects before and after Merge model

From the above tables, some insights are drawn and are as follows:

a) The owner's rights over the object are merged with the beneficiary in both models, whether it is a view, edit or meta-right.
b) In both models, the owner's rights over the parent space remain the same with no change in their previous state.
c) The owner's rights over the child objects are also merged and remain the same in both models.
d) The beneficiary gets the merged rights over the object in both models[136].
e) The beneficiary gets the view right over the parent space in both models.
f) The beneficiary rights over the child object are same for both models.
g) The rights of parent role remain the same in both the models.
h) The rights of offspring role do not change after both the Merge models.
i) The rights of general public role remain the same after both the Merge models.

Based on the above, the definition of rights can be generalized to make it consistent for use rights as well as meta-rights. As the major difference is of the allowed depth of the model

[136] As merging view right gives only view right and merging edit right gives only edit right, and both rights are different but the model is the same, that is, Merge $_{Use}$. The same applies in case of use-rights and meta-rights, where the rights are different but the models' procedure is the same.

along with the meta-rights authority, it is good to include both of them in the rights equation. It will change the rights equation by

$$Right\ (Actor, Object, operation, Depth, Owner) \ldots [eq.\ 7.16]$$

where the first three variables are the same as traditional rights equation. The newly introduced depth relates to the capability of the beneficiary to further allocate a right or not. If it is 0, the actor cannot further allocate the right, while if it is 1, the actor is allowed to further pass the right on. The last variable is the owner of the object and its modification is associated with the allocation of meta-rights over the object.

This generalized equation can be used for the Merge model and can be represented as

$$Right\ (\{JBen1\ \&\ JBen2\}, O, view, 0, Owner) \ldots [eq.\ 7.17]$$

where 0 mentions that it is the merge of use-right and the owner is still the actual authority over the object. Also the merge of meta-rights with the joint owner can be represented as

$$Right\ (Ben, O, view, 1, \{PO\ \&JO\}) \ldots [eq.\ 7.18]$$

7.2.3.2 Precedence analysis

Merge model has lower precedence over other rights allocation models. A rights allocation model has high precedence if two different allocations sequences *x* and *y* ending at the same model achieve the same final state with respect to rights as well as characteristics.

The rights output of Merge model does depend on the order of applications of previous rights allocation models. Thus it has low precedence as the right's state of two different allocation sequences ending at Merge are different, which can be seen in table 7.1 and 7.3, and summarized in table 7.21.

Use-Rights			
Seq. of Model	**Start State**	**Target**	**End State**
Rep → Mrg (B)	*UR({A}, O, opr)*	*A*	*UR({A&B}, O, opr)*
Shr → Mrg (B)	*UR({MO|A}, O, opr)*	*MO, A*	*UR({MO&A&B}, O, opr)*
Rev → Mrg (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO&A}, O, opr)*

Table 7.21: Output Summary of the Merge model

Also, as highlighted in table 7.2 and 7.4, and summarized in table 7.22, if Merge model is applied after any model, it adds its features but also keeps the features of the previous model.

	Replace	Share	Revoke
Mrg	Rep, Mrg	Shr, Mrg	Rev, Mrg

Table 7.22: Precedence of Merge model applications over other rights allocation models

7.2.3.3 Order analysis

The change in order of multiple applications of the Merge model does not affect the final outcome. The order of multiple applications of Merge model does not matter and the output remains the same by changing the order of input. It is visible from tables 7.1 and 7.3, and can also be seen from the following two equations, where

$$Mrg\,(A) \rightarrow Mrg\,(B) \rightarrow Mrg\,(C) = R(\{A\,\&B\,\&\,C\}, O, opr) \ldots [eq.\,7.19]$$

results in the same final stage as

$$Mrg\,(C) \rightarrow Mrg\,(B) \rightarrow Mrg\,(A) = R(\{A\,\&B\,\&\,C\}, O, opr) \ldots [eq.\,7.20]$$

7.3 Completeness

Chapter 3 presented the reduction approach that exhibits all the characteristics of rights allocation found in literature. The reduction, illustrated by a tree structure, was generated by eliminating some of the parameter values arguing that they are not very useful in the current online social interaction scenarios. This section, however, illustrates the completeness of the reduction approach by claiming that it has incorporated all the characteristics and thus the models can support all the other cases as well[137]. The completeness is based on the following axioms:

a) Chain depth deals with the nature of meta-rights, so the meta-rights models cover those cases.
b) Total allocation can be achieved through using multiple-partial allocations.

[137] Even if they are not very useful in current STS scenarios.

c) Some of the parameter values can be achieved through a combination of two or more basic models.

d) The multiple-cardinal mutual exclusive sub-branch is further divided into totality (axiom B), and depth (axiom A). However, the mutual exclusive, multiple-cardinal, partial, jointly-consent, single pass allocation can be achieved using Merge→Replace (axiom C). While the mutual exclusive, multiple-cardinal, partial, several-consent, single pass allocation can be achieved using Replace→Share (axiom C).

e) The single-cardinal, mutual inclusive sub-branch is also covered in multiple-cardinal, mutual inclusive sub-branch as the latter does not put any restriction on the minimum number of beneficiaries.

Referring back to figure 3.3, table 7.23 gives all the possible combinations of the characteristics of rights allocation.

Monotonicity	Cardinality	Totality	Consent	Depth	Model
Mut. Exd.	Singular	Total	Several	Single pass	$\cup\{\text{Replace}_{\text{Use}}\}^*$
Mut. Exd.	Singular	Total	Several	Chain	$\cup\{\text{Replace}_{\text{Meta}}\}^*$
Mut. Exd.	Singular	Total	Joint	Single pass	$\cup\{\text{Replace}_{\text{Use}}\}$
Mut. Exd.	Singular	Total	Joint	Chain	$\cup\{\text{Replace}_{\text{Meta}}\}$
Mut. Exd.	Singular	Partial	Several	Single pass	$\text{Replace}_{\text{Use}}^*$
Mut. Exd.	Singular	Partial	Several	Chain	$\text{Replace}_{\text{Meta}}^*$
Mut. Exd.	Singular	Partial	Joint	Single pass	$\text{Replace}_{\text{Use}}$
Mut. Exd.	Singular	Partial	Joint	Chain	$\text{Replace}_{\text{Meta}}$
Mut. Exd.	Multiple	Total	Several	Single pass	$\cup\{\text{Replace}_{\text{Use}}\rightarrow\text{Share}_{\text{Use}}\}$
Mut. Exd.	Multiple	Total	Several	Chain	$\cup\{\text{Replace}_{\text{Meta}}\rightarrow\text{Share}_{\text{Meta}}\}$
Mut. Exd.	Multiple	Total	Joint	Single pass	$\cup\{\text{Merge}_{\text{Use}}\rightarrow\text{Replace}_{\text{Use}}\}$
Mut. Exd.	Multiple	Total	Joint	Chain	$\cup\{\text{Merge}_{\text{Meta}}\rightarrow\text{Replace}_{\text{Meta}}\}$
Mut. Exd.	Multiple	Partial	Several	Single pass	$\text{Replace}_{\text{Use}}\rightarrow\text{Share}_{\text{Use}}$
Mut. Exd.	Multiple	Partial	Several	Chain	$\text{Replace}_{\text{Meta}}\rightarrow\text{Share}_{\text{Meta}}$
Mut. Exd.	Multiple	Partial	Joint	Single pass	$\text{Merge}_{\text{Use}}\rightarrow\text{Replace}_{\text{Use}}$
Mut. Exd.	Multiple	Partial	Joint	Chain	$\text{Merge}_{\text{Meta}}\rightarrow\text{Replace}_{\text{Meta}}$
Mut. Ind	Singular	Total	Several	Single pass	$\cup\{\text{Share}_{\text{Use}}\}$
Mut. Ind	Singular	Total	Several	Chain	$\cup\{\text{Share}_{\text{Meta}}\}$
Mut. Ind	Singular	Total	Joint	Single pass	$\cup\{\text{Merge}_{\text{Use}}\}$
Mut. Ind	Singular	Total	Joint	Chain	$\cup\{\text{Merge}_{\text{Meta}}\}$

Mut. Ind	Singular	Partial	Several	Single pass	Share $_{Use}$
Mut. Ind	Singular	Partial	Several	Chain	Share $_{Meta}$
Mut. Ind	Singular	Partial	Joint	Single pass	Merge $_{Use}$
Mut. Ind	Singular	Partial	Joint	Chain	Merge $_{Meta}$
Mut. Ind	Multiple	Total	Several	Single pass	$\cup${Share $_{Use}$}
Mut. Ind	Multiple	Total	Several	Chain	$\cup${Share $_{Meta}$}
Mut. Ind	Multiple	Total	Joint	Single pass	$\cup${Merge $_{Use}$}
Mut. Ind	Multiple	Total	Joint	Chain	$\cup${Merge $_{Meta}$}
Mut. Ind	Multiple	Partial	Several	Single pass	Share $_{Use}$
Mut. Ind	Multiple	Partial	Several	Chain	Share $_{Meta}$
Mut. Ind	Multiple	Partial	Joint	Single pass	Merge $_{Use}$
Mut. Ind	Multiple	Partial	Joint	Chain	Merge $_{Meta}$
* These cases may make the system unstable due to the presence of too many unwilling parties					

Table 7.23: All the Possible Options for Various Characteristics of Rights Allocation Framework

The above table shows that the options left in the reduction tree can be covered by the combination of one or more proposed models. However, some of the options may lead the system to an unstable state, which also justify the case why there were left in the first place. It also shows that the designed models are complete and generic enough to cover all the other parameter values that were eliminated while designing the reduction tree for online social interactions.

7.4 Summary

This chapter has analyzed the use-rights and meta-rights allocation models for online social interactions presented in chapter 5 and 6. The possible permutations of the models are outlined to extend the effectiveness of these proposed models and to configure more complex privacy policies using a combination of them. The outcome of these permutations has leaded us to explore some similarities between the models of the same type. The similarities are investigated in values of characteristics of the models, the outcome of their application and their effect on rights of different roles associated with the object, and lead to the generalization of the rights allocation models. The chapter has further investigated the precedence of every rights allocation model over each other, and the effect of multiple application of the same model.

Chapter 8

Demonstration

This chapter demonstrates the rights allocation models presented in previous chapters using both actual and hypothetical use-cases. The model use-cases discuss various practical scenarios from the current STS and suggest their initial configuration using the proposed rights allocation models. However, they will not cover all the functionality of the applications but only the features that are relevant to the access control[138]. This chapter illustrates how specific functions can be implemented in a specific STS using terminologies discussed in this dissertation. Five application types are considered to describe the working of the models and to emphasize the fact that the models are generic enough to accommodate most variations. The chapter is divided into two major sections: a) Basic use-cases, which are similar in most types of applications, and b) Advanced use-cases, which are specific to the application type. Advanced use-cases are both actual and hypothetical – which are not there yet, but the current applications can be enhanced in these directions by using the proposed models. It is significant to mention that both types of use-cases are successfully demonstrated using the proposed rights allocation models.

[138] For example search feature on YouTube (or suggested friends feature on Facebook) is not illustrated.

8.1 Basic use-cases

This section covers the basic use-cases that are present in most types of applications. It is divided with respect to the types of use-cases commonly present.

8.1.1 Creation use-case

Does the system administrator (admin) allow the general public (GP) to create objects (for example profile) in the system space? And who owns that object?

8.1.1.1 Facebook

The Facebook admin allows the GP to create profile in Facebook Space and gives the ownership to the creator.

Rights implementation

$$Replace_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{Owner^{139}\}, FacebookSpace, CreateProfile, 0, admin)$$

8.1.1.2 Orkut[140]

The admin allows the GP to create profile in Orkut Space, and gives the ownership to the creator.

Rights implementation

$$Replace_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{Owner\}, OrkutSpace, CreateProfile, 0, admin)$$

8.1.1.3 YouTube

The YouTube administrator (admin) allows the general public to create/upload some video on YouTube space in their name.

Rights implementation

$$Replace_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{Owner\}, YouTubeSpace, CreateVideo, 0, admin)$$

---

[139] Owner is the actor creating the profile.

[140] Orkut has changed its privacy settings after 2006 and now it is more like Facebook. , Here, the old Orkut system is discussed for giving another perspective and variation in the privacy settings.

8.1.1.4 Wikipedia

The Wikipedia administrator (admin) allows the general public to add some article on Wikipedia space. However, the ownership remains with the system administrator as they can remove it.

Rights implementation

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, WikiSpace, CreateArticle, 0, admin^{141})$$

8.1.1.5 Knowledge management systems

The conference administrator allows the general public to submit some paper to the conference space, where the author is the owner.

Rights implementation

$$Replace_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{Owner^{142}\}, ConferenceSpace, CreatePaper, 0, Owner)$$

8.1.1.6 Discussion

All the above mentioned systems allow the general public to create the object (for example profile, video, article and so on) in the system space. Also, most of the systems give the ownership to the owner but only Wikipedia keeps it with the administrator.

8.1.2 System administrator rights

What are the rights of system administrator (admin) over the newly created object (for example profile, video and so on)?

8.1.2.1 Facebook

The system administrator can view or delete a profile, but cannot edit it.

Rights implementation

$$Share_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{admin|Owner\}, profile, view, 0, Owner)$$
$$Share_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{admin|Owner\}, profile, delete, 0, Owner)$$

---

[141] The artides continue to belong to the admin as they can delete it.

[142] Owner is the actor creating/uploading the paper.

8.1.2.2 Orkut

The system administrator can view or delete a profile, but cannot edit it.

Rights implementation

$$Share_{Use}(\{admin\}, \{Owner|GP\}) \rightarrow UR(\{admin|Owner|GP\}, profile, view, 0, Owner)$$

$$Share_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{admin|Owner\}, profile, delete, 0, Owner)$$

8.1.2.3 YouTube

The system administrator can view or delete a profile/video but cannot edit it.

Rights implementation

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, YouTubeSpace, view, 0, admin)$$

$$Share_{Use}(\{admin\}, \{Owner|GP\}) \rightarrow UR(\{admin|Owner|GP\}, video, view, 0, Owner)$$

$$Share_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{admin|Owner\}, video, delete, 0, Owner)$$

8.1.2.4 Wikipedia

The system administrator can view, edit or delete an article from the space.

Rights implementation

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, WikiSpace, view, 0, admin)$$

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, articles, view, 0, admin)$$

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, articles, edit, 0, admin)$$

$$Share_{Use}(\{admin\}, \{admin\}) \rightarrow UR(\{admin\}, articles, delete, 0, admin)$$

8.1.2.5 Knowledge management system

The system administrator can view or delete a paper, but cannot edit it. They can also allow others (mostly reviewers) to view the paper.

Rights implementation

$$Share_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{admin|Owner\}, paper, view, 0, Owner)$$

$$Share_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{admin|Owner\}, paper, delete, 0, Owner)$$

$$Share_{Meta}(\{admin\}, \{Author\}) \rightarrow MR(\{admin|Author\}, paper, view, 1, Author)$$

8.1.2.6 Discussion

In all the above mentioned cases, the system administrator has the right to delete some object from their space. Even in cases where the ownership resides with the owner, the system administrator shares the delete right with them. This also satisfies the accountability principles as everybody is responsible for their domain and its contents.

8.1.3 General public rights

What are the rights of the general public over the space and the created object (for example profile)?

8.1.3.1 Facebook

The GP gets the view rights in the system space and so can see the basic information about some profile.

Rights implementation

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, FacebookSpace, view, 0, admin)$$

8.1.3.2 Orkut

The GP gets the view rights over the space as well as over the created objects.

Rights implementation

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, OrkutSpace, view, 0, admin)$$
$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, profile, view, 0, Owner)$$

8.1.3.3 YouTube

The GP gets the view rights over the entire space as well as individual videos. The GP also gets the right to vote or comment on the videos.

Rights implementation

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, YouTubeSpace, view, 0, admin)$$
$$Share_{Use}(\{admin|Owner\}, \{GP\}) \rightarrow UR(\{admin|owner|GP\}, video, view, 0, Owner)$$
$$Replace_{Use}(\{admin\}, \{GP|Owner\}) \rightarrow UR(\{GP|Owner\}, video, CreateVote, 0, Owner)$$
$$Replace_{Use}(\{admin\}, \{GP|Owner\}) \rightarrow UR(\{GP|Owner\}, video, CreateComment, 0, Owner)$$

8.1.3.4 Wikipedia

The GP role gets the view and/or edit rights over the entire article space as well as individual articles. The GP also gets the right to vote the authenticity of the articles.

Rights implementation

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, WikiSpace, view, 0, admin)$$

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, articles, view, 0, admin)$$

$$Replace_{Use}(\{admin\}, \{GP|Owner\}) \rightarrow UR(\{GP|Owner\}, article, CreateVote, 0, admin)$$

8.1.3.5 Knowledge management system

The GP does not get any right over the paper, nor can they see the listing.

8.1.3.6 Discussion

As personal data is shared on Facebook, it restricts its visibility. Orkut did not support this and it might be one of the reasons of its relatively slower growth as compared to Facebook[143]. Today, social networks do not give the view right to the general public, while social media (for example YouTube) and social knowledge sharing (for example Wikipedia) allow the GP to explore the whole system space along with the objects in it. However, it is interesting to note that the proposed model can work with all the above mentioned scenarios even in presence of different initialization requirements.

8.1.4 Contents creation

What are the rights of the owner over the newly created objects (for example profile, video, article and so on)?

8.1.4.1 Facebook

The owner can view, edit or delete the profile, they can also decide who can view or contribute to the contents in their personal space.

Rights implementation

$$Share_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{admin|Owner\}, profile, view, 0, Owner)$$

[143] This may be one of the reasons that they changed their visibility function afterwards.

$$Replace_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{Owner\}, profile, edit, 0, Owner)$$
$$Share_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{admin|Owner\}, profile, delete, 0, Owner)$$
$$Replace_{Meta}(\{admin\}, \{Owner\}) \rightarrow MR(\{Owner\}, profile, view, 1, Owner)$$
$$Replace_{Meta}(\{admin\}, \{Owner\}) \rightarrow MR(\{Owner\}, profile, create, 1, Owner)$$

8.1.4.2 Orkut

Owner manages their profile spaces and decides who can contribute to the contents in the space. However, the GP can view the space and the contents in it.

Rights implementation

$$Share_{Use}(\{admin\}, \{Owner|GP\}) \rightarrow UR(\{admin|Owner|GP\}, profile, view, 0, Owner)$$
$$Replace_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{Owner\}, profile, edit, 0, Owner)$$
$$Share_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{admin|Owner\}, profile, delete, 0, Owner)$$
$$Replace_{Meta}(\{admin\}, \{Owner\}) \rightarrow MR(\{Owner\}, profile, create, 1, Owner)$$

8.1.4.3 YouTube

The owner can edit the properties/version of the video, but cannot decide who can comment/vote over their video, nor who can view it. On the contrary, the GP role gets the view rights over the entire space/videos and they can also vote or comment on the videos.

Rights implementation

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, video, view, 0, Owner)$$
$$Replace_{Use}(\{admin\}, \{GP|Owner\}) \rightarrow UR(\{GP|Owner\}, video, CreateVote, 0, Owner)$$
$$Replace_{Use}(\{admin\}, \{GP|Owner\}) \rightarrow UR(\{GP|Owner\}, video, CreateComment, 0, Owner)$$
$$Replace_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{Owner\}, video, edit, 0, Owner)$$

8.1.4.4 Wikipedia

The owner (creator/contributor) does not have the right to decide the allocation of view/edit rights. The GP role gets the view and/or edit rights over the entire article space as well as individual articles. The GP also gets the right to vote the authenticity of the articles.

Rights implementation

$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, WikiSpace, view, 0, admin)$$
$$Share_{Use}(\{admin\}, \{GP\}) \rightarrow UR(\{admin|GP\}, articles, view, 0, admin)$$

$$Replace_{Use}(\{admin\}, \{GP|Owner\}) \rightarrow UR(\{GP|Owner\}, article, CreateVote, 0, admin)$$

8.1.4.5 Knowledge management system

The owner can view, edit or delete the paper, however, they cannot decide about the view rights of viewer or anyone else.

Rights implementation

$$Share_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{admin|Owner\}, paper, view, 0, Owner)$$
$$Replace_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{Owner\}, paper, edit, 0, Owner)$$
$$Share_{Use}(\{admin\}, \{Owner\}) \rightarrow UR(\{admin|Owner\}, paper, delete, 0, Owner)$$

8.1.4.6 Discussion

The owner has more authority on Facebook, as they can restrict the visibility of information, which Orkut[144] does not allow. Social Networks (for example Facebook and Orkut) give the view meta-rights to the owner while social video media (for example YouTube) and social knowledge sharing (for example Wikipedia) allows the whole general public role to view all the space and its objects. However, the proposed model can work with all the above mentioned scenarios even in presence of different content creation conditions.

8.2 Advance use-cases

This section covers the advanced use-cases that are specific to the application type. This section will be divided with respect to the applications and then to various application uses.

8.2.1 Facebook

This section covers the use-cases that are specific to the Facebook application. Facebook is a private owner-oriented application which allows the owner to manage the rights over their created objects. Currently, this type of applications only concentrates on single ownership; however they have the potential to be extended towards multiple ownerships and thus group interactions. The following use-cases cover various options related to the management of rights in Facebook.

[144] Orkut before 2006. Afterwards, Orkut also follows the same logic of local visibility.

8.2.1.1 Friends Rights Management

An actor *A* allows his friends to look at his photo[145] on Facebook. They can also vote or comment on the photo, but could not further allow their friends to look at it.

Rights implementation

$$Share_{Use}(\{A\}, \{Friends\}) \rightarrow UR(\{A|Friends\}, photo, view, 0, A)$$
$$Replace_{Use}(\{A\}, \{Friends\}) \rightarrow UR(\{Friends\}, photo, CreateVote, 0, Friends)$$
$$Share_{Use}(\{A\}, \{Friends\}) \rightarrow UR(\{A|Friends\}, vote, view, 0, Friends)$$
$$Share_{Use}(\{A\}, \{Friends\}) \rightarrow UR(\{A|Friends\}, Photo, CreateComment, 0, A|Friends)$$
$$Share_{Use}(\{A\}, \{Friends\}) \rightarrow UR(\{A|Friends\}, comment, view, 0, Friends)$$
$$Share_{Use}(\{A\}, \{Friends\}) \rightarrow UR(\{A|Friends\}, comment, delete, 0, Friends)$$

8.2.1.2 Tagging a photo

An actor *A* tags another actor *B* on a photo[146]. The rights of both the actors are merged over the photo and both of them required agreeing for any modification of rights allocation over the photo.

Rights implementation

$$Merge_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A\&B\}, photo, view, 1, A\&B)$$
$$Merge_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A\&B\}, photo, edit, 1, A\&B)$$
$$Merge_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A\&B\}, photo, delete, 1, A\&B)$$
$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, photo, view, 0, A\&B)$$
$$Merge_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A\&B\}, photo, edit, 0, A\&B)$$
$$Merge_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A\&B\}, photo, delete, 0, A\&B)$$

8.2.1.3 Sharing a video

An actor *A* shares a video with his friend *B*, who can further share it with his friends. The rights of both the actors are shared over the video and any of them can manage the rights of their friends.

---

[145] The same set of rights can be illustrated for status update, video and other activities posted on one's wall.

[146] Currently this is happening with single ownership of the photo and the owner can display any photo by tagging anyone. However, as tagging means that the photo also showing the other actor, his consent should be taken into account for any display.

Rights implementation

$$Share_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A|B\}, photo, view, 1, A|B)$$
$$Share_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A|B\}, photo, edit, 1, A|B)$$
$$Share_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A|B\}, photo, delete, 1, A|B)$$
$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, photo, view, 0, A|B)$$
$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, photo, edit, 0, A|B)$$
$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, photo, delete, 0, A|B)$$

8.2.1.4 Persona sharing

The possibility of family persona currently does not exist on Facebook. However, with the use of these basic models, husband and wife can own the family personae with their children and have their social circles as relative families and family friends. For this scenario, consider an actor *A* sharing the family personae with another actor *B*, where anyone of them can modify the personae, decide about the friend's requests and other actions associated with the personae.

Rights implementation

$$Share_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A\&B\}, persona, view, 1, A|B)$$
$$Share_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A\&B\}, persona, edit, 1, A|B)$$
$$Share_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A\&B\}, persona, delete, 1, A|B)$$
$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, persona, view, 0, A|B)$$
$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, persona, edit, 0, A|B)$$
$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, persona, delete, 0, A|B)$$

8.2.1.5 Persona merge

The possibility of company/group persona currently does not exist on Facebook. For this scenario, consider an actor *A* as the owner of a company, adding the company board of directors as the joint decision maker.

Rights implementation

$$Merge_{Meta}(\{A\}, \{Directors^{147}\}) \rightarrow MR(\{A\&Directors\}, company, view, 1, A\&Directors)$$
$$Merge_{Meta}(\{A\}, \{Directors\}) \rightarrow MR(\{A\&Directors\}, company, edit, 1, A\&Directors)$$

[147] 'Directors' is the role having all the directors as members.

$$Merge_{Meta}(\{A\}, \{Directors\}) \rightarrow MR(\{A\&Directors\}, company, delete, 1, A\&Directors)$$

$$Share_{Use}(\{A\}, Directors) \rightarrow UR(\{A|Directors\}, company, view, 0, A\&Directors)$$

$$Merge_{Use}(\{A\}, \{Directors\}) \rightarrow UR(\{A\&Directors\}, company, edit, 0, A\&Directors)$$

$$Merge_{Use}(\{A\}, \{Directors\}) \rightarrow UR(\{A\&Directors\}, company, delete, 0, A\&Directors)$$

8.2.1.6 Persona rights delegation

An actor *A*, as the owner of a company, assigns another actor *B* as the acting chairman for the company in his absence. The ownership remains with the original owner, while the use-rights are given to the delegatee.

Rights implementation

$$Replace_{Use}(\{A\}, \{B\}) \rightarrow UR(\{B|Directors\}, company, view, 0, A\&Directors)$$

$$Replace_{Use}(\{A\}, \{B\}) \rightarrow UR(\{B\&Directors\}, company, edit, 0, A\&Directors)$$

$$Replace_{Use}(\{A\}, \{B\}) \rightarrow UR(\{B\&Directors\}, company, delete, 0, A\&Directors)$$

8.2.1.7 Persona transfer

An actor *A*, as the owner of a company, sells his company to another actor *B*, who becomes the new owner.

Rights implementation

$$Replace_{Meta}(\{A\}, \{B\}) \rightarrow UR(\{B|Directors\}, company, view, 0, B\&Directors)$$

$$Replace_{Meta}(\{A\}, \{B\}) \rightarrow UR(\{B\&Directors\}, company, edit, 0, B\&Directors)$$

$$Replace_{Meta}(\{A\}, \{B\}) \rightarrow UR(\{B\&Directors\}, company, delete, 0, B\&Directors)$$

8.2.2 YouTube

This section covers the use-cases that are specific to the YouTube application. YouTube is a fairly public-oriented application and so it does not have many group interaction use-cases. Following are the use-cases covering various options related to the management of rights in YouTube.

8.2.2.1 Video transfer

An actor *A* transfers his video to another actor *B*. The rights of the system administrator as well as general public role remain the same.

Rights implementation

$$Replace_{Use}(\{A\}, \{B\}) \rightarrow UR(\{B\}, video, edit, 0, B)$$
$$Share_{Use}(\{admin\}, \{B|GP\}) \rightarrow UR(\{admin|B|GP\}, video, view, 0, B)$$
$$Share_{Use}(\{admin\}, \{B\}) \rightarrow UR(\{admin|B\}, video, delete, 0, B)$$
$$Replace_{Use}(\{admin\}, \{GP|Owner\}) \rightarrow UR(\{GP|Owner\}, video, create, 0, B)$$

8.2.3 Wikipedia

Wikipedia is a purely public-oriented application, so it is less likely to be extended towards these social interactions. This dissertation has not included any advanced case on Wikipedia use.

8.2.4 Knowledge management system

This section covers the use-cases that are specific to the knowledge management systems. Currently, these systems only provide quite a few functionalities but they have the potential to extend towards group interactions. Following are some of the use-cases that cover various options related to the management of rights in these systems.

8.2.4.1 Track delegation

The conference chair *A* delegates a track to track chair *B* and give him full control over the track. The track chair can view and edit the track on the conference chair behalf but cannot delete it. Also the track chair cannot further pass that right to someone else.

Rights implementation

$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, track, view, 0, A)^{148}$$
$$Replace_{Use}(\{A\}, \{B\}) \rightarrow UR(\{B\}, track, edit, 0, A)$$
$$Replace_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{B\}, track, view, 0, A)$$

8.2.4.2 Paper joint authorship

An author *A* adds another actor *B* as one of the authors of the paper, where both of them required agreeing for any update or removal of the paper.

Rights implementation

[148] The conference chair A still remains the owner of the conference and track.

$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, paper, view, 0, A\&B)$$
$$Merge_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A\&B\}, paper, edit, 0, A\&B)$$
$$Merge_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A\&B\}, paper, delete, 0, A\&B)$$
$$Merge_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A\&B\}, paper, view, 1, A\&B)$$
$$Merge_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A\&B\}, paper, edit, 1, A\&B)$$
$$Merge_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A\&B\}, paper, delete, 1, A\&B)$$

8.2.4.3 Paper authorship sharing

An author *A* shares the authorship with another actor *B*, where anyone of them can perform any update on the paper.

Rights implementation

$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, paper, view, 0, A|B)$$
$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, paper, edit, 0, A|B)$$
$$Share_{Use}(\{A\}, \{B\}) \rightarrow UR(\{A|B\}, paper, delete, 0, A|B)$$
$$Share_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A|B\}, paper, view, 1, A|B)$$
$$Share_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A|B\}, paper, edit, 1, A|B)$$
$$Share_{Meta}(\{A\}, \{B\}) \rightarrow MR(\{A|B\}, paper, delete, 1, A|B)$$

8.2.4.4 Reviewer rights

The conference chair (*admin*) gives the view rights over the paper to the reviewers X, Y and Z.

Rights implementation

$$Share_{Use}(\{admin\}, \{X|Y|Z\}) \rightarrow UR(\{admin|X|Y|Z\}, paper, view, 0, Owner)$$

8.2.4.5 Copyright transfer

The paper author *A*, transfers the copyright of the view rights over the paper to conference chair (*admin*).

Rights implementation

$$Replace_{Meta}(\{A\}, \{admin\}) \rightarrow MR(\{admin\}, paper, view, 1, A)$$

8.3 Summary

This chapter has demonstrated the use-cases that are covered by the proposed rights allocation models. The use-cases are categorized into a) basic and b) advanced; where basic use case are commonly present in most of the current applications supporting online social interactions, and advanced use-cases are specific to the application type. Some interesting observations have been made as two entirely different types of applications (YouTube and Wikipedia) provide quite similar types of rights allocation support. Also, the set of functions within an application increases as more rights are allocated to the owners. It supports the Locke argument that societies that encourage free will and ownership prosper more.

The proposed models give options to various STS and cover their current functionality. Also, few hypothetical cases are also discussed to illustrate the further extension of these application towards becoming more group oriented. The illustrations also help to estimate the full potential of the proposed models where it supports not only most of the current scenarios but also the online group social interactions, which is not present yet.

Chapter 9

Discussion

This chapter discusses the research questions and the contributions of this dissertation. It also outlines some of its limitations and illustrates some future directions. It is organized as follows: section 9.1 revisits the research questions along with its research contributions in section 9.2. Section 9.3 provides some theoretical and practical implications, and section 9.4 discusses some of the limitations of the presented work. Section 9.5 gives some future directions particularly in the proposed work and generally in the access control for online social interactions, while section 9.6 formally ends the dissertation.

This research was motivated by the absence of a consistent and broadly applicable rights allocation model for online social interactions, and the differences between the structure of STS and traditional applications which make them difficult to be used for online social interactions. The objective of this research was to investigate the possibility of designing standard rights allocation models for online social interactions. This objective was twofold: a) to design the basic access control model supporting ownership, relationship and local control; and b) to design the rights allocation models for online social interactions based on the semantics of basic access control model. The research has achieved both the above mentioned objectives by presenting the social access control model in chapter 4 and by presenting the rights allocation models in chapter 5 and 6.

SAC model has been proposed as the basic access control model for STS. It supports the STS requirements of ownership, relationship, and local administration. These three components allow the enforcement of heterogeneous privacy policies with ease and reduced overhead. The model illustrates how systems with huge number of users can be divided into autonomous distributed domains. These owner administered domains permit the introduction of local roles, attestation certificates and object classes, where the former two allow the management of dynamic asymmetric relationships and the latter contributes towards refined resource management with reduced overhead. These constructs not only satisfy the technical requirements of efficiency and scalability but also the social requirements of ownership, relationships and heterogeneous privacy policies. The latter arises because online contents are contributed by the community of users, who by social logic expect to own what they contribute. The research has used socio-technical design to reduce the socio-technical gap. The aim of this design is to satisfy both technical and social needs to avoid technical and social errors, with community outrage is an example of the latter. The logical model is defined to work with any existing security module.

The SAC model was then used as the core model to build the rights allocation models. The rights allocation models are Replace, Share, Merge and Revoke which were outlined for use rights as well as meta-rights, to result in the formation of eight different models. These models were discussed using various current online social interaction scenarios to illustrate their need and practical importance. Furthermore, some of the characteristics of rights allocation were discussed along with the rights of different roles associated with the owner domain, which help to identify the logical structure of each model. The models were further generalized on the basis of their features and some analysis is done to understand the generalized models.

9.1 Revisiting research questions

This section demonstrates the scope and success of the research by revisiting the research questions and describing how they were answered.

Q) Can a general rights allocation model for online social interactions, which is decentralized, logically consistent, socially valid, and supports dynamic local roles, be designed?

The rights allocation models are presented in chapters 5 and 6. The models were mapped onto the social access control model to show that they are decentralized, support dynamic local roles associated with stakeholder's namespace, and socially valid by its support for ownership, freedom, privacy, relationships and local administration. The models were designed to be logically consistent as shown by their shift from different roles and their access towards different objects before and after allocation of right.

a) What are the basic types of rights allocation for* online social interactions*?

There are four basic types of rights allocation discussed in chapter 3, which were later elaborated in chapters 5 and 6. The allocation models are Replace, Share, Merge and Revoke, and they were mapped on to simple rights and meta-rights giving eight models in total. The complete discussions of these models are given in chapters 5 and 6.

b) Can the allocation models be applied to simple rights as well as meta-rights?

These use-rights allocation models are presented in chapter 5. The Replace $_{Use}$ model is presented in section 5.1, the Share $_{Use}$ model is presented in section 5.2, while the Merge $_{Use}$ model is presented in section 5.3. The Revoke model is discussed separately under each of the above mentioned models in section 5.1.6, 5.2.6 and 5.3.6. The meta-rights allocation models are presented in chapter 6. The Replace $_{Meta}$ model is presented in section 6.1, the Share $_{Meta}$ model is presented in section 6.2, while the Merge $_{Meta}$ model is presented in section 6.3. The Revoke model is discussed separately under each of the above mentioned models in section 6.1.6, 6.2.6 and 6.3.6.

c) What are the characteristics of those rights allocations?

Various characteristics of rights allocation are extracted from literature and their parameters have been adjusted using socio-technical design. The characteristics associated with each type of allocation are consent, totality, cardinality, monotonicity and depth. These characteristics are defined in general in chapter 3 and discussed with respect to each specific model in chapter 5 and 6.

d) What are the rights of various roles associated with an object after every allocation?

To answer this question, various generalized roles are identified for every object space like parent, offspring and general public in chapter 4 (section 4.3) along with their basic rights over the owner object. Additionally, the modification in the rights for every role after each allocation model is discussed in chapter 5 and 6. Further, the generalized rights states for these roles are discussed: a) for the Replace model – in section 7.1.3, b) for the Share model – in section 7.2.3, and c) for the Merge model – in section 7.3.3.

e) What is the difference between the allocation models for simple and meta-rights?

After critical analysis of the use-rights and meta-rights models, chapter 7 concludes that the prominent differences between use-rights and meta-rights allocation models are the depth of the allocated right and the revocation by meta-owner. So the models are generalized to work with use rights as well as meta-rights without modifying the semantics of the models. This generalization is discussed in chapter 7 for various characteristics of the model as well as for the rights associated with every role.

Condition) Can an access control model for STS be designed to support ownership, relationships and local administration?

Chapter 4 of this dissertation has presented SAC – the core distributed access control model for STS, which supports ownership, relationship and local administration. The model divides the whole system into autonomous domains, with the distinction of stake holders, virtual users, local roles, attestation certificates and object privacy classes. The Stakeholder is the administrative authority for the local domain, who manages the local roles, attestation certificates, and object classes. The Virtual user is a delegate pointer to the actual user's persona and is used to reduce the complexity of information management. Local roles are domain based groups, whose membership is based on relationships. Attestation certificates are encapsulated rights and are used with local roles to support dynamic asymmetric relationships. Object classes are introduced to ease the administration of contents and their mapping to appropriate local roles. This provides the facility to enforce heterogeneous privacy policies through local control over domain and its components. The complete description of the SAC model is given in chapter 4.

a) Can the model be applied to emerging scenarios of* online social interactions*?

The proposed SAC model is applied throughout to various scenarios of current STS in chapter 5 and 6. Before outlining the characteristics of every rights allocation model, an online social interaction scenario is depicted using the SAC model. The model is illustrated through conference systems, YouTube, collaborative software, Facebook and internet forums. These demonstrations illustrates that SAC can be used for most types of STS without any modification in their semantics.

9.2 Research contributions

This research has presented Social Access Control model for online social interactions based on the social principles of ownership, relationships and local administration. Further, it has explored various ways to allocate a right in STS, and proposed their logical semantics. Some innovative contributions of this research are summarized as follows:

a) The concept of domains is used for online social interactions without any formal semantics based on local requirements and programmer's intuition, so there are many variations of it. The SAC model formalizes the owner oriented domains to support ownership and local administration. It defines various components of the domain and explores their interaction.
b) Fine grained social circles were proposed in (J. Li et al., 2009; Tapiador et al., 2011), but they do not support the dynamic nature of friendship. The SAC model not only provides full control to the owner over modification of relationship, but also formalizes the notion of fine grained social circles.
c) The object tagging was explored in (Hart et al., 2007), but it was based on rules rather than owner preferences. The SAC model has introduced owner oriented object privacy classes to support object classification based on their contents and owner privacy policy.
d) SAC has introduced attestation certificates in STS, which also contributed towards decentralized access control credential distribution. These certificates provide a mechanism to support asymmetric relationships.

e) SAC has introduced initial space configuration to support different types of online application using the basic social access control model. These configurations allow the SAC model to be used for most types of online applications supporting social interactions.

f) The research has used socio-technical design in the area of access control research, and proposed an access control model based on the social structure of ownership, relationship and local control.

g) It has explored various characteristics of rights allocations and introduced a reduction tree illustrating useful options in current online social interaction scenarios. The reduction tree has outlined the logical choices for online interactions, and has shown to be complete in order to cover all the remaining possible options.

h) Rights delegation was introduced in (Gasser & McDermott, 1990) for local and remote processes. It was matured to the delegation of roles in (Barka & Sandhu, 2004). However, there does not exist any rights delegation model for ownership domain or STS. This research has introduced a rights delegation model for online social interactions based on ownership and local administration. This will enhance online interactions by providing backups and decentralization of authority.

i) Rights transfer for organizational roles was introduced in (Barka, 2000) and refined in (Crampton & Khambhammettu, 2008). However, STS are entirely different from organizations and there does not exist any rights transfer model for online social interactions based on ownership. This research has introduced a rights transfer model for online social interactions based on ownership and local administration. This will enhance the online ownership framework and may introduce more complex ideas like selling/purchasing of online objects and personae.

j) The concept of use-rights and meta-rights sharing is used in current STS without any formal semantics based on programmers' intuition, so there are many variations of it[149]. This research has formalized the rights sharing supporting ownership and local control. This formalization provides the programmers with a refined set of rules and programmable instructions which can be used for most types of online applications

[149] From video sharing on social networks to photo tagging (where one can remove their id to remove the access right of their social circle).

supporting social interactions. It also gives the opportunity to introduce the ownership sharing of various interesting online objects like family personae and family friends.

k) It has introduced the rights merge model for online social interactions. The proposed model supports ownership and local control and enhances collaborative scenarios, team work and distribution of authority, and thus improves online social interactions. It also gives the opportunity to introduce group interactions in the management of rights like company personae with board of directors as joint owners.

l) The research has used extreme formal modeling in the area of access control research to formalize the rights allocation models. It allows the introduction of fine grained models based on the specifications of owner oriented domains, local roles and attestation certificates.

m) This research has outlined the four basic allocation models for use-rights as well as for meta-rights to extend the availability of rights in ownership domain. The detailed outlining of these models along with their logical definitions provide the programmers with exact semantics for the desired application. The introduction of these allocation models will improve the online social interactions and offer much richer options to online users.

9.3 Implications of the presented research

This research has several implications on the theory of access control and on the current state of the STS software system development. This section will cover those implications.

9.3.1 Ownership theory

This research supports the ownership theory associated with objects (Locke, 1963) and extend it to include the administration of personal relationships and their rights over one's resources. The access control for STS works on the basic theory of ownership, which states that

"The owner has the meta-right over an object to administer it".

The proposed access control model supports this theory of ownership and extends it towards relationship management. The model considers the relationship between two actors as a type of directed information object owned by the owner. This creates two directed relationship

objects for a single relationship between *Alice* and *Bob*, one owned by *Alice* for how *Bob* relates to her, and one owned by *Bob* for how *Alice* relates to him. The extended theory includes:

"Relationship is also an information object which is owned by the owner."

9.3.2 Access control

This research summarizes the literature of access control model for STS and explores that they have the common grounds of ownership and relationships. It also adds the observation that socio-technical design can be applied to the access control domain so it can work with the social structure of the community. This research has introduced roles in the discretionary domain and discussed the opportunities associated with it.

This research has made an interesting observation that meta-rights are special types of rights associated with the administration of rights. However, when it comes to the rights allocation models, the same models that are used for use rights can also be used for the allocation of meta-rights. The only difference between the two types of models is the depth and the revocation of a right, which are related to the nature of meta-rights and do not affect the standard procedure of rights allocation.

9.3.3 Online social interactions

This research extends the online social interactions from individual users to groups. It permits families, groups and organization to have their combined personae and their online interactions. This can introduce various interesting opportunities for objects sharing, online and offline relationships, and make online social interactions more realistic and closer to the real world.

Group online personae will let cities, and countries to have their personae which can help improving the governance of offline societies and the relations of different groups with each other. It may also encourage discussion of political situations with the perspective of citizens and nationals, and may help in resolving contradictory issues playing the role of online United Nations.

9.4 Limitations

Following are some of the limitations of this research:

a) The research does not provide any mechanism or architecture to implement the rights allocation models. Also, the simulation was not done due to the scope of this research[150]. If it was taken into account the whole debates of client server architectures, distributed control, heterogeneous privacy policies, users' behavior, their feedback and some other concerns also need to be considered. However, the research provides the logical formulae and the logical steps that can be directly translated into any programming language.
b) The basic access control model is based on ownership, relationship and local control, but has not considered trust between the users, and reputation of a user in the community.
c) The models have assumed that every actor is authorized and there is no identity theft, while it is not the scope of this research to deal with these matters, these factors can affect the real world implementation.
d) The similarities between different roles with in a namespace are not taken into account which can significantly reduce the computational overhead and storage efficiency of access control.

9.5 Future research opportunities

This section discusses the future research opportunities particularly in the proposed models, and generally in the access control for STS. Following are some of the future research opportunities:

9.5.1 Implementation

A future direction can be advised to implement the proposed rights allocation models as a component of the security kernel for STS and include all the proposed options to the users. The implementation can take advantage of semantic web ontologies for its inter-reference qualities and can use Google stream as the simulation agent. This simulation and

[150] It only deals with the design of the allocation models.

implementation may be able to suggest solutions to some of the other interesting debates posed by the current research, for example whether centralized or distributed implementation architecture is more suitable for STS; and whether client-side or server-side management of policy credentials is better for load management in STS. Also the implementation can be tested against various network attacks to suggest an error resilient approach suitable for STS. Additionally as the content retrieval for a user is based on their social circle, the implementation may give insights about storing data in a more efficient way. The implementation would be an interesting addition to the literature of security in general.

9.5.2 Transparency

Another interesting research direction in access control for STS is the introduction of transparency, so users know about what to expect rather than making social errors. In general, transparency is the choice to view rights that affect you. Transparency in access control provides readable error messages for guidance to what users can do, in terms of allowed rights. In social terms, transparency of access control rules allow users to anticipate and avoid social errors and reduces community governance corruption as people see the permissions of others (Kooiman, Bavinck, Chuenpagdee, Mahon, & Pullin, 2008). The goal is to show that social rights are openly applied and can be scrutinized, as this is critical for trust and synergy. If a social requirement of access control systems is transparency, this sets access control apart from the security aim of system defence, which by definition requires secrecy. The evolution of access control to meet the needs of social networks opens it up to new research dimensions beyond its security origins.

Some design options for transparency model are: a) the model should be able to generate statements that actor *X* has permission *P* over object *O*; b) before putting any object into a space, the object owner may sign a contract with the space owner that they have such rights over the object and parent space; c) upon entering any space the model should notify the user that they have such rights over the space and the objects within it. These design options may help in designing a transparency model that may translate the possible actions of the security kernel.

9.5.3 Reputation model

Another interesting research direction for access control for STS can be the reputation model. The model may be able to calculate the reputation of a user in a community by how trustworthy the community considers the user. The reputation model for access control may suggest the owner to opt for a particular rights allocation model[151] based on the reputation of the requestor. The reputation model that rates users in a community can have multiple research implications: a) the model can be used by online shopping systems, which tracks the reputation of a particular user; b) it can be used by online trading systems to rate someone's trustworthiness to trade; c) a reputation based access control model can be designed to allow access to resources based on the reputation of the user.

In the context of this research, the reputation model for rights allocation models will determine whether a particular user is trustworthy enough for delegating/transferring of rights or not. The model may be based on some reputation rating system calculated on user's previous transactions. The model may introduce an automatic type of trust evaluator that provides suggestions as to which allocation model is best suited for users with different trust levels. The model needs to be distributed, dynamic in nature and flexible enough to suggest the level of delegation/transfer based on the requestor's reputation. However, due to the dynamic nature of online social interactions and heterogeneous users' policies, designing a reputation model presents an interesting challenge to the research community.

9.6 Final remarks

Online communities today cannot survive without participation, so access control is increasingly about letting people in rather than keeping them out. This research suggests how to allocate access control rights to satisfy social requirements of ownership and relationship. It not only defines what STS like Facebook currently should and should not does, but also suggests new options not yet tried. The proposed model helps to avoid social errors at source thus increasing the probability of online social success. The evolution of access control towards the allocation of meta-rights will open up new research dimensions.

[151] This particular user is better suited for rights delegation rather than rights transfer.

If the Internet is to be a global community, it must agree on a consistent logic of online social rights. This research offers steps to the development of a standard and consistent rights allocation model for all online social interactions. However, rights logic is powerful but complex, as people can form groups, objects can contain other objects and rights can overlap and contradict, for example free speech is not the right to defame. A socio-technical designer might wonder, if even legal theorists cannot agree on social rights, how can we cope? Yet some justice is always better than none, whether online or off. To do nothing until perfect justice is defined is not how social evolution occurs.

This research has described, in access control terms, social rights in online social interactions. This opens the way for the development of social standards for the internet, just as it already has standards for hardware, software and HCI. Such standards would agree on how code should mediate online social interactions in order to improve social performance by social principles like synergy, fairness, creator ownership and transparency. Socio-technical systems need social standards of rights to reach their full potential.

"You may never know what results come of your action, but if you do nothing there will be no result"

(~Mahatma Gandhi)

Glossary

Access Control	–	A process to grant certain privileges over information and resources to identified users.
Actor	–	A subject/user/person/group in the context of access control model.
Attestation certificates (AC)	–	Permission objects encapsulated various access rights and map local role to object class.
Beneficiary	–	The actor who is allocated some right.
Cardinality	–	Cardinality of an allocation refers to the number of beneficiaries, who can simultaneously hold an allocated right.
Consent	–	This characteristic deals with whether the consent of the owner and beneficiary is required for an allocation or not.
Constructive research methodology	–	The constructive methodology is a research approach for producing innovative constructions, intended to solve problems faced by real world and thus makes contributions to the theory of the applied discipline.
Creator Ownership	–	A right that one should own what one creates, for example a painting or poem.
Discretionary access control (DAC)	–	Access control model that is based on personal discretion and users can give rights to other users.
Delegatee (Dge)	–	The beneficiary of the use-right where the owner cannot exercise

the right. They are the only accountable actor for the delegated right over the object.

Delegator (Dgr) – The owner of the delegated right, who cannot exercise the right without taking it back from the delegatee.

Denial of Service (DOS) – An attack launched on the web services to deny that service to the authorized users.

Depth – This characteristic refers to the ability of the beneficiary after getting a right to further allocate it or not.

Extreme Formal Modeling (XFM) – Extreme Formal Modeling is a methodology that constructs abstract models which are constructively correct in nature and closer to intended specification.

Freedom – Freedom is the right to control one's body, to not be a slave to another.

General Public (GP) – A general role defined for each space and has some rights over the objects present in that space.

Incremental approach – The approach where one feature is focused at a time. Multiple life cycles take place, where one is completed before starting the next.

Joint Beneficiary – The beneficiary who cannot exercise the right without the consent of the owner.

Joint Owner – The owner of an object who cannot exercise the right over the owned-object without the consent of the (primary) owner.

Local roles (LR) – A VU group with defined access to NS resources.

Mandatory access control (MAC) – A centralized access control model where the system administrator is responsible for assigning rights.

Merge – Merge the rights of the entire user set over an object, so all parties must agree to perform some action.

Meta-Rights – Meta-rights are system permissions for actors to apply operations on rights.

Monotonicity – This characteristic refers to the state of rights of previous actors after allocation.

Multiple-ownership – When multiple actors hold the ownership of an object.

Namespaces (NS) – The set of objects a stakeholder creates.

Object Classes (OC) – An object group, based on security clearance, whose access is mapped to LRs.

Object – The system resources that needs to be protected.

Offspring – The owner of the child object in perspective of the owner of the space.

Online Social Interaction – Online Social Interactions are user to user social interactions that take place in STS. It is a concept that is used to explain different types of interactions among users of a system. STS are systems supporting OSI.

Operation – Applicable actions on objects/actors/rights.

Owner – The actor having all the use-rights and meta-rights over an object

Parent – The owner of the parent space, where the object is created / uploaded. It is a role from the perspective of the owner of the object.

Persona – An online persona represents an offline party, for example an avatar, profile, or a mail account.

Primary Owner – The actual owner of an object who has shared / merged some rights over their object with secondary / joint owners.

Replace – Give rights to another user who can exercise them on the owner's behalf, where the previous user cannot exercise it.

Revocation – Revocation is a process by which rights are taken back from the beneficiary.

Revoke – Taking back the rights over an object from an actor. It can be done by removing the existing actor from a right over an object.

Right – A logical triplet that defines the actor who can perform an operation over an object.

Rights Allocation – Methods in which a right can be assigned to different users.

Role based access control (RBAC) – An access control model based on organizational roles, where the security administrator assigns the roles and permissions to users.

Secondary Owner – The owner of an object who can exercise the right without the consent of the (primary) owner.

Share – Give away rights over an object while keeping them at the same

time, so both can exercise the rights.

Simple rights – Use-rights are system permissions for actors to apply operations on objects.

Social Access control – The access control model that supports the social requirements/ axioms of the online social interactions and is proposed in this research.

Social Networks – Social Networks are systems supporting friendships and relationships. They are subset of STS and are one of the richest examples of online social interactions.

Socio-technical System (STS) – A socio-technical system (STS) is a social system sitting upon a technical base, with email a simple example of social communication by technology means. They allow people to communicate with each other through technology rather than through physical means. It is used in this dissertation to discuss systems that support online social interactions.

Stakeholder – The owner who posts system resource objects, for example photos, videos, comments or votes in his namespace.

Totality – A characteristic that deals with whether the complete set of rights over complete set of authorized resources are allocated.

Use-Rights – Use-rights are system permissions for actors to apply operations on objects.

Virtual user (VU) – A user, from the social circle of stakeholder, seeking a NS resource access.

References

Abadi, M. (2009). Logic in Access Control. In A. Aldini, G. Barthe, & R. Gorrieri (Eds.), Foundations of Security Analysis and Design (pp. 145-165).

Abadi, M., Burrows, M., Lampson, B., & Plotkin, G. (1993, September). A Calculus for Access Control in Distributed Systems. ACM Transactions on Programming Languages and Systems, 15(4), 706-734.

Adams, W. J., & Davis, N. J. (2005). Toward a decentralized trust-based access control system for dynamic collaboration. IEEE Workshop on Information Assurance and Security, (pp. 317-324). New York, USA.

Ajzen, I. (1985). From intentions to actions: A theory of planned behavior. (J. Kuhl, & J. Beckman, Eds.) Action-control: From cognition to behavior, 11-39.

Ajzen, I. (1991). The theory of planned behavior. Organizational Behavior and Human Decision Processes, 50, 179-211.

Ali, B., Villegas, W., & Maheswaran, M. (2007). A trust based approach for protecting user data in social networks. 18th Annual International Conference on Computer Science and Software Engineering (CASCON '07), (pp. 288-293). Richmond-Hill, Ontario, Canada.

Allen, A. (1988). Uneasy Access: Privacy for Women in a Free Society. Totowa, New Jersey, USA: Rowman and Littlefield Publishers.

Amati, G., & Crestani, F. (1999). Probabilistic learning for selective dissemination of information. Information Processing and Management, 35(5), 633-654.

Atluri, V., & Warner, J. (2005). Supporting conditional delegation in secure workflow management systems. 10th ACM symposium on Access control models and technologies, (pp. 49-58). Stockholm, Sweden.

Ba, S. (2001). Establishing online trust through a community responsibility system. Decision Support Systems, 31(3), 323-336.

Bammigatti, P. H., & Rao, P. R. (2008). Delegation in role based access control model for workflow systems. International Journal of Computer Science and Security, 2(2), 1-10.

Barka, E. S. (2002). Framework for Role-Based Delegation Models. PhD Dissertation. Virginia, USA: George Mason University.

Barka, E., & Sandhu, R. (2000a). A Role-Based Delegation Model and Some Extensions. 23rd National Information Systems Security Conference (NISSC), (pp. 101-114). Baltimore, Maryland, USA .

Barka, E., & Sandhu, R. (2000b). Framework for Role-Based Delegation Models. 16th Annual Computer Security Applications Conference (ACSAC), (pp. 168-176). New Orleans, Louisiana, USA.

Barka, E., & Sandhu, R. (2004). Role-Based Delegation Model/ Hierarchical Roles (RBDM1). 20th Annual Computer Security Applications Conference (ACSAC) (pp. 396-404). Tucson, Arizona, USA: IEEE Computer Society.

Barka, E., & Sandhu, R. (2007). Framework for Agent-Based Role Delegation. IEEE International Conference on Communications (ICC), (pp. 1361-1367). Glasgow, Scotland.

Barnes, S. (2006). Privacy paradox: Social networking in the United States. First Monday, 11(9).

Belani, E., Vahdat, A., Anderson, T., & Dahlin, M. (1998). The CRISIS wide area security architecture. 7th conference on USENIX Security Symposium, (pp. 15-30). Berkeley, USA.

Bell, E., & LaPadula, L. (1973). Secure Computer Systems: Mathematical Foundations and Model. MITRE Technical Report 2547.

Bender, T. (1978). Community and social change in America. Princeton, New Jersey, USA: Rutgers University Press.

Bennett, P., Dumais, S., & Horvitz, E. (2002). Probabilistic combination of text classifiers using reliability indicators: Models and results. 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 207-214). Tampere, Finland: ACM Press.

Bertalanffy, L. V. (1968). General System Theory. New York, USA: George Braziller Inc. Publishers.

Bertino, E., Fan, J., Ferrari, E., Hacid, M.-S., Elmagarmid, A., & Zhu, X. (2003). A hierarchical access control model for video database systems. ACM Transactions on Information Systems, 21(2), 155-191.

Biba, K. (1977). Integrity Considerations for Secure Computer Systems. USA: MITRE Corporation.

Blau, P. M. (1964). Exchange and Power in Social Life. New York, USA: Willey.

Bock, G.-W., & Kankanhalli, A. (2006). Are Norms Enough? The Role of Collaborative Norms in Promoting Organizational Knowledge Seeking. European Journal of Information Systems, 15(4), 357-367.

boyd, d. (2006). Friends, Friendsters, and MySpace Top 8: Writing Community Into Being on Social Network Sites. First Monday, 11(12).

boyd, d. M., & Ellison, N. B. (2007). Social Network Sites: Definition, History, and Scholarship. Journal of Computer-Mediated Communication, 13(1), 210-230.

Boykin, P. O., & Roychowdhury, V. P. (2005, April). Leveraging social networks to fight spam. IEEE Computer Magazine, 38(4), 61-68.

Bram, C. (2003). Incentives build robustness in BitTorrent. 1st Workshop on Economics of Peer-to-Peer Systems. Berkeley.

Breen, G. (2010, May 17th). ABC News. Retrieved March 23rd, 2011, from ABC News website: http://www.abc.net.au/news/2010-05-17/teens-murder-sparks-facebook-privacy-plea/829850

Brewer, D., & Nash, M. (1989). The Chinese Wall Security Policy. IEEE Computer Society Symposium on Research in Security and Privacy, (pp. 215-228). Oakland, California, USA.

Brucker, A., & Petritsch, H. (2009). Extending Access Control Models with Break-glass. Symposium on Access Control Models and Technologies (SACMAT), (pp. 197-206). Stresa, Italy.

Burrell, G., & Morgan, G. (1979). Sociological Paradigms and Organisational Analysis: Element of the Sociology of Corporate Life. Heinemann, London, UK: Heinemann Educational Books.

Caplinskas, A., & Vasilecas, O. (2004). Information systems research methodologies and models. International Conference on Computer Systems and Technologies (CompSysTech), (pp. 1-6). Rousse, Bulgaria.

Carminati, B., & Ferrari, E. (2008). Privacy-aware collaborative access control in web-based social networks. 22nd annual IFIP WG 11.3 working conference on Data and Applications Security (pp. 81-96). London, UK: Springer-Verlag Berlin, Heidelberg.

Carminati, B., Ferrari, E., & Perego, A. (2006). Rule-based access control for social networks. On the Move to Meaningful Internet Systems (OTM) Workshops (pp. 1734-1744). Springer-Verlag Berlin Heidelberg.

Carminati, B., Ferrari, E., & Perego, A. (2007). Private relationships in social networks. 23rd IEEE International Conference on Data Engineering Workshop, (pp. 163-171). Istanbul, Turkey.

Carminati, B., Ferrari, E., & Perego, A. (2009). Enforcing access control in Web-based social networks. ACM Transactions on Information and System Security (TISSEC), 13 (1), 191-233.

Carminati, B., Ferrari, E., Heatherly, R., Kantarcioglu, M., & Thuraisingham, B. (2009). A semantic web based framework for social network access control. 14th ACM Symposium on Access Control Models and Technologies, (pp. 177-186). New York, USA.

Carminati, B., Ferrari, E., Heatherly, R., Kantarcioglu, M., & Thuraisingham, B. (2011). Semantic web-based social network access control. Computers and Security Journal, 30(2), 108-115.

Choi, H.-C., Kruk, S. R., Grzonkowski, S., Stankiewicz, K., Davis, B., & Breslin, J. G. (2006). Trust models for community aware identity management. Identity, Reference, and the Web Workshop (IRW). Edinburgh, Scotland.

Chu, Y.-H., Feigenbaum, J., Lamacchia, B., Resnick, P., & Strauss, M. (1997). Referee: Trust management for web applications. World Wide Web Journal, 706-734.

Chung, M., Choi, J., Lee, K., & Rhyoo, S.-K. (2007). Enterprise Application Framework for Constructing Secure RFID Application. International Conference on Advanced Language Processing and Web Information Technology (ALPIT), (pp. 572-577). Luoyang, Henan, China.

Churcharoenkrung, N., Kim, Y. S., & Kang, B. H. (2005). Dynamic web content filtering based on user's knowledge. International Conference on Information Technology: Coding and Computing, (pp. 184-188). Hobart, Tasmania, Australia.

Clark, D., & Wilson, D. (1987). A Comparison of Commercial and Military Computer Security Policies. IEEE Symposium on Security and Privacy, (pp. 184-194). Oakland, California, USA.

Clegg, C. (2000). Sociotechnical principles for system design. Applied Ergonomics, 31(5), 463-477.

Coiera, E. (2007). Putting the technical back into socio-technical systems research. International Journal of Medical Informatics, 76, 98-103.

Collins, M., & Singer, Y. (1999). Unsupervised models for named entity classification. joint SIGDAT conference on empricial methods in natural language processing and very large corpora, (pp. 100-110). Maryland, USA.

Crampton, J., & Khambhammettu, H. (2006). Delegation in role-based access control. 11th European Symposium on Research in Computer Security (pp. 174-191). Hamburg, Germany: Springer-Verlag.

Crampton, J., & Khambhammettu, H. (2008). Delegation in role-based access control. International Journal of Information Security, 7(2), 123-136.

Crnkovic, G. D. (2010). Constructive Research and Info-Computational Knowledge Generation. Model-Based Reasoning in Science and Technology, 359-380.

Davenport, D. (2009). The Social Derivation of Technical Systems. In B. Whitworth, & A. de Moor (Eds.), Handbook of Research on Socio-Technical Design and Social Networking Systems (pp. 50-64). Hershey, Pennsylvania, USA: IGI Global.

Deelmann, T., & Loos, P. (2002). Trust Economy: Aspects of Reputation and Trust Building for SMEs in E-business. 8th Americas Conference on Information Systems, (pp. 2213-2221). Dallas, Texas, USA.

Dewan, P., & Shen, H. (1998). Flexible meta access-control for collaborative applications. ACM conference on Computer supported cooperative work, (pp. 247-256). Seatle, Washington, USA.

Diamond, J. (1999). Guns, Germs and Steel: The Fates of Human Societies. London: Vintage.

Dimmock, N., Belokosztolszki, A., Eyers, D., Bacon, J., & Moody, K. (2004). Using trust and risk in role-based access control policies. 9th ACM symposium on Access control models and technologies (SACMAT), (pp. 156-162). New York, USA.

Edjlali, G., Acharya, A., & Chaudhary, V. (1998). History-Based Access Control for Mobile Code. 5th ACM conference on Computer and communications security, (pp. 38-48). San Francisco, California, USA.

Edjlali, G., Acharya, A., & Chaudhary, V. (1999). History-Based Access Control for Mobile Code. Secure Internet Programming, Security Issues for Mobile and Distributed Objects. Lecture Notes in Computer Science, 413-431.

Elahi, N., Chowdhury, M. M., & Noll, J. (2008). Semantic access control in web based communities. International Multi-Conference on Computing in the Global Information Technology (ICCGI) (pp. 131-136). Washington, DC, USA: IEEE.

Encyclopedia of Philosophy, S. (2007). Rights.

Etzioni, A., & Etzioni, O. (1999). Fact-to-face and computer-mediated communities, a comparative analysis. The Information Society, 15(4), 241-248.

Fan, Y.-W., & Wu, C.-C. (2011). The Role of Social Capital in Knowledge Sharing: A Meta-Analytic Review. 44th Hawaii International Conference on System Sciences (HICSS), (pp. 1-10). Hawaii, USA.

Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, R., & Chandramouli, R. (2001, August). Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and Systems Security (TISSEC), 4(3), 224-274.

Ferraiolo, D., & Kuhn, R. (1992). Role-Based Access Control. 15th NIST-NSA National Computer Security Conference, (pp. 554-563). Baltimore, Maryland, USA.

Ferraiolo, D., Gilbert, D., & Lynch, N. (1993). An Examination of Federal and Commercial Access Control Policy Needs. 16th NIST-NSA National Computer Security Conference, (pp. 107-116). Baltimore, Maryland, USA.

Ferraiolo, D., Kuhn, D. R., & Chandramouli, R. (2003). Role Based Access Control. Artech House.

Fishbein, M., & Ajzen, I. (1975). Belief, attitude, intention and behavior: An introduction to theory and research. Addison-Wesley Publishing Company.

Fong, P. W., Anwar, M., & Zhao, Z. (2009). A Privacy Preservation Model for Facebook-Style Social Network Systems. 14th European conference on Research in computer security (pp. 303-320). Saint Malo, France: Springer-Verlag Berlin, Heidelberg.

Freeden, M. (1991). Rights (Concepts in Social Thought). Minnesota, USA: University of Minnesota Press.

Freudenthal, E., Pesin, T., Port, L., Keenan, E., & Karamcheti, V. (2002). dRBAC: Distributed role-based access control for dynamic coalition environments. 22nd International Conference on Distributed Computing Systems, (pp. 411-420). New York, USA.

Fung, R., & Lee, M. (1999). EC-trust (trust in electronic commerce): exploring the antecedent factors. 15th Americas Conference on Information Systems (pp. 517-519). Milwaukee, Wisconsin, USA: AIS.

Gaaloul, K., Schaad, A., Flegel, U., & Charoy, F. (2008). A secure task delegation model for workflows. 2nd International Conference on Emerging Security Information, Systems and Technologies, (pp. 10-15). Cap Esterel, France.

Gasser, M., & McDermott, E. (1990). An Architecture for practical Delegation in a Distributed System. IEEE Computer Society Symposium on Research in Security and Privacy, (pp. 20-30). Oakland, California, USA.

Gavison, R. (1980). Privacy and the Limits of Law. The Yale Law Journal, 89(3), 421-471.

Gladney, H. M. (1997, April). Access Control for Large Collections. ACM Transactions on Information Systems, 15(2), 154-194.

Glass, R., Ramesh, V., & Vessey, I. (2004). An Analysis of Research in Computing Disciplines. Sprouts: Working Papers on Information Systems, 47(6), 89-95.

Goguen, J. A., & Linde, C. (1993). Techniques for Requirements Elicitation. International Symposium on Requirements Engineering, (pp. 152-164). San Diego, USA.

Golbeck, J. (2009). Trust and nuanced profile similarity in online social networks. ACM Transactions on the Web (TWEB) Volume 3, Issue 4.

Gollu, K. K., Saroiu, S., & Wolman, A. (2007). A Social Networking-Based Access Control Scheme for Personal Content. 21st ACM Symposium on Operating Systems Principles (SOSP). Stevenson, Washington, USA.

Gregoriades, A., Shin, J.-E., & Sutcliffe, A. (2004). Human-centred requirements engineering. 12th IEEE International Requirements Engineering Conference, (pp. 154-163). Kyoto, Japan.

Grudin, J. (1990). The Computer Reaches Out: The historical continuity of user interface design. SIGCHI Conference on Human Factors in Computing Systems, (pp. 261-268). Seattle, Washington, USA.

Guo, H., & Georganas, N. D. (2002). Digital Image Watermarking for Joint Ownership. 10th ACM Confaence on Multimedia, (pp. 362-371). Juan les Pins, France.

Hart, M., Johnson, R., & Stent, A. (2007). More Content- Less Control: Access Control in the Web 2.0. Workshop on Web 2.0 Security and Privacy at the IEEE Symposium on Security and Privacy. Oakland, California, USA.

Heuvel, W. J. (2002). Integrating Modern Business Applications with Objectified Legacy Systems. PhD Thesis. Center for Economic Research, Tilburg University.

Hippel, E. v. (2005). Democratizing Innovation. Cambridge, USA: The MIT Press.

Iannella, R. (2001). Digital Rights Management (DRM) Architectures. D-Lib Magazine, 7(6).

Ilic, A., Michahelles, F., & Fleisch, E. (2007). The Dual Ownership Model: Using Organizational Relationships for Access Control in Safety Supply Chains. 21st International Conference on Advanced Information Networking and Applications Workshops, (pp. 459-466). Niagara Falls, Canada.

Indratmo, & Vassileva, J. (2007). A Usability Study of an Access Control System for Group Blogs. International Conference on Weblogs and Social Media (ICWSM), (pp. 235-238). Boulder, Colorado, USA.

Jahnke, I. (2009). Socio-Technical Communities: From Informal to Formal? In B. Whitworth, & A. de Moore (Eds.), Handbook of Research on Socio-Technical Design and Social Networking Systems (pp. 763-778). Hershey, Pennsylvania, USA: IGI Global.

Jajodia, S., & Sandhu, R. (1991). Toward a multilevel secure relational data model. ACM International Conference on Management of Data (SIGMOD), (pp. 50-59). Denver, Colorado, USA.

Joshi, J., & Bertino, E. (2006). Fine-grained role-based delegation in presence of the hybrid role hierarchy. 11th ACM symposium on Access control models and technologies (SACMAT), (pp. 81-90). California, USA.

Kalman, M., Monge, P., Fulk, J., & Heino, R. (2002). Motivations to Resolve Communication Dilemmas in Database-Mediated Collaboration. Communication Research, 29, 125-154.

Kamvar, S. D., Schlosser, M. T., & Garcia-Molina, H. (2003). The eigentrust algorithm for reputation management in p2p networks. 12th international conference on World Wide Web, (pp. 640-651). New York, USA.

Kane, K. M. (2006). Access Control in Decentralized, Distributed Systems. PhD Dissertation. University of Texas, Austin.

Karp, A., Haury, H., & Davis, M. (2010). From ABAC to ZBAC: the evolution of access control models. 5th International Conference on Information Warfare and Security, (pp. 1-21). Ohio, USA.

Kasanen, E., Lukka, K., & Siitonen, A. (1993). The constructive approach in management accounting research. Journal of Management Accounting Research, 5, 241-264.

Kate, S. t. (2009). Trustworthiness within social networking sites: A study on the intersection of hci and sociology. Amsterdam, Netherlands: University of Amsterdam.

Kerschbaum, F. (2010). An Access Control Model for Mobile Physical Objects. 15th ACM symposium on Access control models and technologies, (pp. 193-202). New York, USA.

Kim, D., Song, Y., Braynov, S., & Rao, R. (2001). A B-to-C Trust Model for Online Exchange. 7th Americas Conference on Information Systems, (pp. 784-787). Boston, USA.

Kim, E., & Tadisina, S. (2003). Customers' Initial Trust in E-Businesses: How to Measure Customers' Initial Trust. 9th Americas Conference on Information Systems, (pp. 35-41). Tampa, Florida, USA.

Kim, Y.-H., Hahn, S.-Y., & Zhang, B.-T. (2000). Text filtering by boosting naive bayes classifiers. 23rd annual international ACM SIGIR conference on Research and development in information retrieval, (pp. 168-175). Athens, Greece.

Kirkpatrick, M., & Bertino, E. (2010). Enforcing Spatial Constraints for Mobile RBAC Systems. Symposium on Access Control Models and Technologies (SACMAT), (pp. 99-108). Pittsburgh, Pennsylvania, USA.

Kling, R., & Lamb, R. (1999). IT and Organizational Change in Digital Economies: A Socio-Technical Approach. ACM SIGCAS Computers and Society, 29(3), 17-25.

Kling, R., McKim, G., & King, A. (2003). A bit more to it: Scholarly Communication Forums as Socio-Technical Interaction Networks. Journal of the American Society for Information Science and Technology, 54(1), 47-67.

Knoke, D. (1986). Associations and interest groups. Annual Review of Sociology, 12, 1-21.

Kooiman, J., Bavinck, M., Chuenpagdee, R., Mahon, R., & Pullin, R. (2008). Interactive governance and governability: an introduction. The Journal of Transdisciplinary Environmental Studies, 7(1).

Kraft, D., & Schafer, G. (2004). Distributed access control for consumer operated mobile ad-hoc networks. IEEE Consumer Communications and Networking Conference, (pp. 35-40). Las Vegas, USA.

Kruk, S. R., Grzonkowski, S., Gzella, A., Woroniecki, T., & Choi, H.-C. (2006). D-FOAF: Distributed identity management with access rights delegation. Asian Semantic Web Conference (ASWC) (pp. 140-154). Beijing, China: LNCS, Springer Verlag.

Kuhn, T. (1970). The Structure of Scientific Revolutions. Chicago, USA: Chicago University Press.

Kulkarni, D., & Tripathi, A. (2008). Context-Aware Role-based Access Control in Pervasive Computing Systems. Symposium on Access Control Models and Technologies (SACMAT), (pp. 113-122). Estes Park, Colorado, USA.

Kuutti, K. (1996). Activity Theory as a Potential Framework for Human Computer Interaction Research. In B. A. Nardi (Ed.), Context and Consciousness: Activity Theory and Human-Computer Interaction (pp. 9-22). Cambridge, Massachusetts, USA: The MIT Press.

Lamb, R., & Kling, R. (2003). Re-conceptualizing users as social actors in information systems research. MIS Quarterly, 27(2), 197-235.

Lampson, B. (1972). Protection and access control in operating systems. Operating Systems, Infotech State of the Art Report, 309-326.

Lampson, B. W. (1969). Dynamic protection structures. American Federation of Information Processing Societies (AFIPS), (pp. 27-38). New Jersey, USA.

Lee, H., Lee, K., & Chung, M. (2006). Enterprise application framework for constructing secure RFID application. International Conference on Hybrid Information Technology, (pp. 480-489). Jeju Island, Korea.

Lessig, L. (1999). Code and other laws of cyberspace. New York, USA: Basic Books.

Li, J., Tang, Y., Mao, C., Lai, H., & Zhu, J. (2009). Role Based Access Control for Social Network Sites. Joint Conferences on Pervasive Computing (JCPC), (pp. 389-394). Taipei, Taiwan.

Li, N., & Grosof, B. (2000). A Practically Implementable and Tractable Delegation Logic. IEEE Symposium on Security and Privacy, (pp. 27-42). Berkeley, California, USA.

Li, N., Feigenbaum, J., & Grosof, B. (1999). A logic-based knowledge representation for authorization with delegation. 12th international IEEE Computer Security Foundations Workshop (CSFW), (pp. 162-174). Mordano, Italy.

Li, N., Grosof, B., & Feigenbaum, J. (2003). Delegation Logic: A logic-based approach to distributed authorization. ACM Transactions of Information and System Security (TISSEC), 6(1), 128-171.

Linn, J., & Nystrom, M. (1999). Attribute Certification: An Enabling Technology for Delegation and Role-Based Controls in Distributed Environments. ACM Workshop on RBAC, (pp. 121-130). Fairfax, Virginia, USA.

Liu, H., Lim, E.-P., Lauw, H. W., Le, M.-T., Sun, A., Srivastava, J., et al. (2008). Predicting trusts among users of online communities: an epinions case study. 9th ACM conference on Electronic commerce, (pp. 310-319). Chicago, Illinois, USA.

Liu, Q., Safavi-Naini, R., & Pau, N. (2003). Digital Rights Management for Content Distribution. Australasian information security workshop conference on ACSW frontiers, (pp. 49-58). Adelaide, Australia.

Liu, Y., Yu, N., & Hao, Z. (2009). Rights Sharing Scheme for Online DRM System Using Digital Ticket. International Conference on Management and Service Science (MASS), (pp. 1-6). Hefei, China.

Locke, J. (1963). An essay concerning the true original extent and end of civil government: Second of "Two Treatises on Government. In J. Somerville, & R. Santoni (Eds.), Social and Political Philosophy: Readings From Plato to Gandhi (pp. 169-204). Anchorp.

Lukka, K. (2003). The constructive research approach. In O. Lauri, & O.-P. Hilmola (Eds.), Case study research in logistics (pp. 83-101).

Majchrzak, A., Wagner, C., & Yates, D. (2006). Corporate Wiki Users: Results of a Survey. Iinternational symposium on Wikis, (pp. 99-104). Odense, Denmark.

Massa, P., & Avesani, P. (2007). Trust-aware recommender systems. ACM conference on Recommender systems, (pp. 17-24). Minneapolis, Minnesota, USA.

Matsuo, Y., & Yamamoto, H. (2009). Community gravity: measuring bidirectional effects by trust and rating on online social networks. 18th international conference on World wide web, (pp. 751-760). Madrid, Spain.

Mattas, A., Mavridis, I., Ilioudis, C., & Pagkalos, I. (2006). Dynamic access control administration for collaborative applications. 10th WSEAS international conference on Computers, (pp. 355-360). Athens, Greece.

Maturana, H. R., & Varela, F. J. (1998). The Tree of Knowledge: the biological roots of human understanding. Boston: Shambhala Publications.

Mayer, R. C., Davis, J. H., & Schoorman, F. D. (1995, July). An Integrative Model of Organizational Trust. The Academy of Management Review, 20(3), 709-734.

McInnerney, J. M., & Roberts, T. S. (2004). Online Learning: Social Interaction and the Creation of a Sense of Community. Educational Technology & Society, 7(3), 73-81.

McKnight, D. H., Choudhury, V., & Kacmar, C. (2000). Trust in E-Commerce Vendors: A Two-Stage Model. 21st International Conference on Information Systems, (pp. 532-536). Brisbane, Australia.

Meo, P. D., Nocera, A., Quattrone, G., Rosaci, D., & Ursino, D. (2009). Finding reliable users and social networks in a social internetworking system. International Database Engineering & Applications Symposium, (pp. 173-181). Cetraro, Calabria, Italy.

Miller, S., Neuman, C., Schiller, J. I., & Saltzer, J. H. (1987). Kerberos Authentication and Authorization System. Cambridge, Massachusetts, USA: Project Athena Technical Plan, MIT Project Athena.

Mills, H. D., Dyer, M., & Linger, R. C. (1987). Cleanroom Software Engineering. IEEE Software, 4(5), 19-25.

Mingers, J. (2001). Combining IS Research Methods: Towards a Pluralist Methodology. Information Systems Research, 12(3), 240-259.

Moore, A. D. (2003). Privacy: Its Meaning and Value. American Philosophical Quarterly, 40(3), 215-227.

Morchon, O. G., & Wehrle, K. (2010). Modular context-aware access control for medical sensor networks. 15th ACM symposium on Access control models and technologies (SACMAT), (pp. 129-138). Pittsburgh, Pennsylvania, USA.

Munawer, Q. (2000). Administrative Models For Role-Based Access Control. PhD Dissertation. Virginia, USA: George Mason University.

Myers, B. A. (1998, April). A brief history of human-computer interaction technology. Interactions, 5(2), 44-54.

Na, S., & Cheon, S. (2000). Role delegation in role-based access control. 5th ACM workshop on Role-based access control, (pp. 39-44). Berlin, Germany.

Oliver, R. L. (1980, November). A Cognitive Model of the Antecedents and Consequences of Satisfaction Decisions. Journal of Marketing Research, 17(4), 460-469.

Oltsik, J. (2009). The Network Security Architecture (NSA) Meets Web 2.0. USA: The Enterprise Strategy Group.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web. Stanford, California, USA: Stanford InfoLab.

Park, D.-G., & Lee, Y.-R. (2005). A Flexible Role-Based Delegation Model Using Characteristics of Permissions. 16th international conference on Database and Expert Systems Applications (pp. 310-323). Springer-Verlag Berlin, Heidelberg.

Park, J. (2003). Usage Control: A Unified Framework For Next Generation Access Control. PhD Dissertation. Virginia, USA: George Mason University.

Pavlou, P. A., Tan, Y.-H., & Gefen, D. (2003). The Transitional Role of Institutional Trust in Online Interorganizational Relationships. 36th Annual Hawaii International Conference on System Sciences (HICSS), (pp. 215-224). Hawaii, USA.

Poulsen, K. (2006, October 16). MySpace predator caught by code. Retrieved July 31, 2011, from Wired: http://www.wired.com/science/discoveries/news/2006/10/71948?currentPage=all

Preece, J., & Shneiderman, B. (2009). The Reader-to-Leader Framework: Motivating Technology-Mediated Social Participation. AIS Transactions on Human-Computer Interaction, 1(1), 13-32.

Pujol, J. M., Sanguesa, R., & Delgado, J. (2002). Extracting Reputation in Multi Agent System by means of Social Network Topology. International Joint Conference on Autonomous Agents and Multi-Agent Systems (pp. 467-474). Bologna, Italy: ACM.

Putnam, R. (2000). Bowling alone: The collapse and revival of American community. New York, USA: Simon and Schuster.

Qian, X., & Lunt, T. (1996). A MAC policy framework for multilevel relational databases. IEEE Transactions on Knowledge and Data Engineering, 8(1), 3-15.

Rand, A., & Branden, N. (1964). Virtue of Selfishness. New York, USA: Penguin Books.

Ridings, C., & Gefen, D. (2004). Virtual Community Attraction: Why People Hang Out Online. Journal of Computer-Mediated Communication, 10(1).

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM, 21(2), 120-126.

Roloff, M. E. (1981). Interpersonal Communication: The Social Exchange Approach. Beverly Hills, California, USA: Sage Publications.

Rose, E. A. (2000). Balancing internet marketing needs with consumer concerns: a property rights framework. ACM SIGCAS Computers and Society, 30(2), 20-24.

Rothaermel, F. T., & Sugiyama, S. (2001). Virtual internet communities and commercial success: individual and community-level theory grounded in the atypical case of TimeZone.com. Journal of Management, 27, 297-312.

Rowse, D. (2006, July 2). http://www.problogger.net/archives/2006/02/07/blog-stalkers-personal-safety-for-bloggers. Retrieved July 31, 2011, from Problogger: http://www.problogger.net/archives/2006/02/07/blog-stalkers-personal-safety-for-bloggers/

Ruan, C., & Varadharajan, V. (2010). A graph theoretic approach to authorization delegation and conflict resolution in decentralised systems. Distributed and Parallel Databases, 27(1), 1-29.

Samarati, P., & Vimercati, S. D. (2001). Access Control: Policies, Models, and Mechanisms. In R. Focardi, & R. Gorrieri (Eds.), Foundations of Security Analysis and Design (pp. 137-196). Springer-Verlag.

Samarati, P., Bertino, E., & Jajodia, S. (1996, August). An authorization model for a distributed hypertext system. IEEE Transactions on Knowledge and Data Engineering, 8(4), 555-562.

Sanders, M., & McCormick, E. J. (1993). Human Factors in Engineering and Design. New York, USA: McGraw-Hill.

Sandhu, R., & Chen, F. (1998). The multilevel relational (MLR) data model. ACM Transactions on Information and Systems Security (TISSEC), 1(1), 93-132.

Sandhu, R., & Munawer, Q. (1998a). How to do discretionary access control using roles. ACM workshop on Role-based access control, (pp. 47-54). New York, USA.

Sandhu, R., & Munawer, Q. (1998b). The RRA97 Model for Role-Based Administration of Role Hierarchies. 14th Annual Computer Security Applications Conference (ACSAC, (pp. 39-49). Scottsdale, Arizona, USA.

Sandhu, R., & Munawer, Q. (1999). The ARBAC99 Model for Administration of Roles. 15th Annual Computer Security Applications Conference (ACSAC), (pp. 229-238). Phoenix, Arizona, USA.

Sandhu, R., & Samarati, P. (1994). Access Control: Principles and Practice. Communications Magazine, IEEE, 40 - 48.

Sandhu, R., Bhamidipati, V., & Munawer, Q. (1999, February). The ARBAC97 Model for Role-Based Administration of Roles. ACM Transactions on Information and Systems Security (TISSEC), 2(1), 105-135.

Sandhu, R., Coyne, E., Feinstein, H., & Youman, C. (1996, February). Role-based access control model. IEEE Computer, 29(2), 38-47.

Scacchi, W. (2004). Socio-technical design. In W. S. Bainbridge (Ed.), The Encyclopedia of Human Computer Interaction. Berkshire Publishing Group.

Sebastiani, F. (2002, March). Machine learning in automated text categorization. ACM Computing Surveys (CSUR), 34(1), 1-47.

Shamir, A. (1979, November). How to share a secret. Communications of the ACM, 22(11), 612-613.

Shapiro, D. L., Sheppard, B. H., & Cheraskin, L. (1992, October). Business on a Hankshake. Negotiation Journal, 8(4), 365-377.

Simonetti, E. (2004, December 16). I was fired for blogging. Retrieved July 31, 2011, from CNet: http://news.cnet.com/i-was-fired-for-blogging/2010-1030_3-5490836.html

Simpson, A. (2008). On the need for user-defined fine-grained access control policies for social networking applications. Workshop on Security in Opportunistic and Social networks, (pp. 1-8). New York, USA.

Sommerville, I. (2004). Software engineering (7th ed.). Addison-Wesley.

Sproull, L., & Arriaga, M. (2007). Online Communities. In H. Bidgoli (Ed.), Handbook of Computer Networks: Volume 3, Distributed Networks, Network Planning, Control, Management, and New Trends and Applications. New Jersey, USA: John Wiley & Sons, Inc.

Steffen, R., & Knorr, R. (2005). A Trust Based Delegation System for managing Access Control. Advances in Pervasive Computing: Adjunct Proceedings Pervasive, (pp. 1-5). Munich, Germany.

Steiner, J. G., Neuman, C., & Schiller, J. I. (1988). Kerberos: An Authentication Service for Open Network Systems. Usenix Winter Conference, (pp. 191-202). Dallas, Texas, USA.

Suhaib, S., Mathaikutty, D., Shukla, S., & Berner, D. (2005). XFM: Extreme formal method for capturing formal specification into abstract models. ACM Transactions on Design Automation of Electronic Systems (TODAES), 589-609.

Tamassia, R., Yao, D., & Winsborough, W. H. (2004). Role-based casdaded delegation. ACM Symposium on Access Control Models and Technologies (SACMAT), (pp. 146-155). New York, USA.

Tapiador, A., Carrera, D., & Salvachúa, J. (2011). Tie-RBAC: an application of RBAC to Social Networks. Web 2.0 Security and Privacy (W2SP). Oakland, California.

Tashakkori, A., & Teddlie, C. (1998). Mixed Methodology: Combining Qualitative and Quantitative Approaches. Thousand Oaks, California, USA: Sage.

TCSEC. (1985). Trusted Computer Security Evaluation Criteria. USA: Department of Defense.

Thompson, M., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K., & Essiari, A. (1999). Certificate-based access control for widely distributed resources. 8th conference on USENIX Security Symposium, (pp. 215-228). Berkeley, USA.

Tran, H., Hitchens, M., Varadharajan, V., & Watters, P. (2005). A trust based access control framework for p2p file-sharing systems. 38th Annual Hawaii International Conference on System Sciences (HICSS), (p. 302c). Honolulu, Hawaii, USA.

Trist, E. L. (1981). The evolution of socio-technical systems : A conceptual framework and an action research program.

Trist, E. L., Higgin, G. W., Murray, H., & Pollock, A. B. (1963). Organizational choice: Capabilities of groups at the coal face under changing technologies. London: Tavistock Publications.

Vanetti, M., Binaghi, E., Carminati, B., Carullo, M., & Ferrari, E. (2010). Content-based Filtering in On-line Social Networks. International ECML/PKDD Workshop on Privacy and Security Issues in Data Mining and Machine Learning (PSDML), (pp. 127-140). Barcelona, Spain.

Varadharajan, V., Allen, P., & Black, S. (1991). An Analysis of the Proxy Problem in Distributed systems. IEEE Symposium on Research in Security and Privacy, (pp. 255-175). Oakland, California, USA.

Wainer, J., & Kumar, A. (2005). A fine-grained, controllable, user-to-user delegation method in RBAC. 10th ACM symposium on Access control models and technologies (SACMAT), (pp. 59-66). Stockholm, Sweden.

Welke, R. (1981). DBMS Support for information systems development. Hamilton, Ontario, Canada.: McMaster University.

Wellman, B., & Gulia, M. (1999). Virtual communities as communities. In M. Smith, & P. Kollock (Eds.), Communities in Cyberspace (pp. 167-194). New York, USA: Routledge.

Whitworth, B. (2009). The Social Requirements of Technical Systems. In B. Whitworth, & A. deMoor (Eds.), Handbook of Research on Socio-Technical Design and Social Networking Systems (pp. 3-22). Hershey, Pennsylvania, USA: IGI Global.

Whitworth, B. & Ahmad, A. (2012). Socio-Technical System Design. In M. Soegaard, & R. F. Dam (Eds.), Encyclopedia of Human-Computer Interaction. Aarhus, Denmark: The Interaction-Design.org Foundation.

Whitworth, B., & deMoor, A. (2002). Legitimate by design: Towards trusted virtual community environments. 35th Hawaii International Conference on System Sciences, (pp. 2831-2842). Hawaii, USA.

Whitworth, B., & deMoor, A. (2003). Legitimate by design: Towards trusted virtual community environments. Behaviour and Information Technology, 31-51.

Whitworth, B., & Friedman, R. (2009a, August). Reinventing academic publishing online Part I: Rigor, Relevance and Practice. First Monday, 14(8).

Whitworth, B., & Friedman, R. (2009b, September). Reinventing academic publishing online Part II: A Socio-technical Vision. First Monday, 14(9).

Whitworth, B., deMoor, A., & Liu, T. (2006). Towards a Theory of Online Social Rights. international conference on On the Move to Meaningful Internet Systems, LNCS 4277 (pp. 247-256). Montpellier, France: Springer-Verlag Berlin Heidelberg.

Wu, M.-Y., Ke, C.-K., & Tzeng, W.-L. (2008). Applying context-aware RBAC to RFID security management for application in retail business. IEEE Asia-Pacific Services Computing Conference, (pp. 1208-1212). Yilan, Taiwan.

Wu, Y. (2003). Dynamic Ownership Verification. 4th International Conference on Information, Communications and Signal Processing, (pp. 985-988). Singapore.

Ye, C., Wu, Z., & Fu, Y. (2005). An Attribute-Based-Delegation-Model and Its Extension. 3rd International Workshop on Security in Information Systems (WOSIS), (pp. 146-159). Miami, USA.

Ye, C., Wu, Z., & Fu, Y. (2006). An Attribute-Based Delegation Model and Its Extension. Journal of Research and Practice in Information Technology, 38(1), 3-17.

Yoon, S.-J. (2002). The antecedents and consequences of trust in online-purchase decisions. Journal of Interactive Marketing, 16(2), 47-63.

Zhang, L., Ahn, G.-J., & Chu, B.-T. (2001). A Rule-Based Framework for Based Delegation. ACM Symposium on Access Control Models and Technologies, (pp. 153-162). Chantilly, Virginia, USA.

Zhang, L., Ahn, G.-J., & Chu, B.-T. (2003, August). A rule-based framework for role-based delegation and revocation. ACM transactions on information and system security, 6(3), 404-441.

Zhang, X., & Zhang, Q. (2005). Online trust forming mechanism: approaches and an integrated model. 7th International conference on Electronic commerce (pp. 201-209). Xi'an, China: ACM.

Zhang, X., Oh, S., & Sandhu, R. (2003). PBDM: A Flexible Delegation Model in RBAC. 8th ACM Symposium on Access Control Models and Technologies (SACMAT), (pp. 149-157). Villa Gallia, Como, Italy.

Zhang, Z., Pei, Q., Ma, J., Yang, L., & Fan, K. (2008). A Fine-Grained Digital Rights Transfer Policy and Trusted Distribution and Enforcement. International Conference on Computational Intelligence and Security , (pp. 457-462). Suzhou, China.

Appendix A

Table 7.1

Below is table 7.1, illustrating use-rights allocation sequences, referred by section 7.1.1

Use-Rights			
Allocation Sequence	**Start State**	**Target**	**End State**
Replace			
Rep (A)	*UR({MO}, O, opr)*	*MO*	*UR({A}, O, opr)*
Rep → Shr (B)	*UR({A}, O, opr)*	*A*	*UR({A|B}, O, opr)*
Rep → Shr → Mrg (C)	*UR({A|B}, O, opr)*	*A,B*[152]	*UR({A&B&C}, O, opr)*
Rep → Shr → Mrg → Rev (C)	*UR(A&B&C}, O, opr)*	*C*	*UR({A&B}, O, opr)*
Rep → Shr → Rev (B)	*UR({A|B}, O, opr)*	*B*	*UR({A}, O, opr)*
Rep → Shr → Rev → Mrg (B)	*UR({A}, O, opr)*	*A*	*UR({A&B}, O, opr)*
Rep → Mrg (B)	*UR({A}, O, opr)*	*A*	*UR({A&B}, O, opr)*
Rep → Mrg → Rev (B)	*UR({A&B}, O, opr)*	*B*	*UR({A}, O, opr)*
Rep → Mrg → Rev → Shr (B)	*UR({A}, O, opr)*	*A*	*UR({A|B}, O, opr)*
Rep → Mrg → Shr (C)	*UR({A&B}, O, opr)*	*A,B*	*UR({A&B&C}, O, opr)*
Rep → Mrg → Shr → Rev (C)	*UR({A&B&C}, O, opr)*	*C*	*UR({A&B}, O, opr)*
Rep → Rev (A)	*UR({A}, O, opr)*	*A*	*UR({MO}, O, opr)*
Rep → Rev → Shr (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO|A}, O, opr)*
Rep → Rev → Shr → Mrg (B)	*UR({MO|A}, O, opr)*	*MO, A*	*UR({MO&A&B}, O, opr)*

[152] If only *B* is the target actor, then the result would be *UR({A|(B&C)}, O, opr)*.

Rep → Rev → Mrg (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO&A}, O, opr)*
Rep → Rev → Mrg → Shr (B)	*UR({MO&A}, O, opr)*	*MO, A*	*UR({MO&A&B}, O, opr)*
Share			
Shr (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO|A}, O, opr)*
Shr → Mrg (B)	*UR({MO|A}, O, opr)*	*MO, A*	*UR({MO&A&B}, O, opr)*
Shr → Mrg → Rev (B)	*UR({MO&A&B}, O, opr)*	*B*	*UR({MO&A}, O, opr)*
Shr → Mrg → Rev → Rep (B)	*UR({MO&A}, O, opr)*	*A*	*UR({MO&B}, O, opr)*
Shr → Mrg → Rep (C)	*UR({MO&A&B}, O, opr)*	*B*	*UR({MO&A&C}, O, opr)*
Shr → Mrg → Rep → Rev (C)	*UR({MO&A&C}, O, opr)*	*C*	*UR({MO&A}, O, opr)*
Shr → Rev (A)	*UR({MO|A}, O, opr)*	*A*	*UR({MO}, O, opr)*
Shr → Rev → Rep (A)	*UR({MO}, O, opr)*	*MO*	*UR({A}, O, opr)*
Shr → Rev → Rep → Mrg (B)	*UR({A}, O, opr)*	*A*	*UR({A&B}, O, opr)*
Shr → Rev → Mrg (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO&A}, O, opr)*
Shr → Rev → Mrg → Rep (B)	*UR({MO&A}, O, opr)*	*A*	*UR({MO&B}, O, opr)*
Shr → Rep (B)	*UR({MO|A}, O, opr)*	*A*	*UR({MO|B}, O, opr)*
Shr → Rep → Mrg (C)	*UR({MO|B}, O, opr)*	*MO,B*	*UR({MO&B&C}, O, opr)*
Shr → Rep → Mrg → Rev (C)	*UR({MO&B&C}, O, opr)*	*C*	*UR({MO&B}, O, opr)*
Shr → Rep → Rev (B)	*UR({MO|B}, O, opr)*	*B*	*UR({MO}, O, opr)*
Shr → Rep → Rev → Mrg (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO&A}, O, opr)*
Merge			
Mrg (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO&A}, O, opr)*
Mrg → Rev (A)	*UR({MO&A}, O, opr)*	*A*	*UR({MO}, O, opr)*
Mrg → Rev → Rep (A)	*UR({MO}, O, opr)*	*MO*	*UR({A}, O, opr)*
Mrg → Rev → Rep → Shr (B)	*UR({A}, O, opr)*	*A*	*UR({A|B}, O, opr)*
Mrg → Rev → Shr (A)	*UR({MO}, O, opr)*	*A*	*UR({MO|A}, O, opr)*
Mrg → Rev → Shr → Rep (B)	*UR({MO|A}, O, opr)*	*A*	*UR({MO|B}, O, opr)*
Mrg → Rep (B)	*UR({MO&A}, O, opr)*	*A*	*UR({MO&B}, O, opr)*
Mrg → Rep → Shr (C)	*UR({MO&B}, O, opr)*	*MO,B*	*UR({MO&B&C}, O, opr)*
Mrg → Rep → Shr → Rev (C)	*UR({MO&B&C}, O, opr)*	*C*	*UR({MO&B}, O, opr)*
Mrg → Rep → Rev (B)	*UR({MO&B}, O, opr)*	*B*	*UR({MO}, O, opr)*
Mrg → Rep → Rev → Shr (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO|A}, O, opr)*
Mrg → Shr (B)	*UR({MO&A}, O, opr)*	*MO,A*	*UR({MO&A&B}, O, opr)*
Mrg → Shr → Rev (B)	*UR({MO&A&B}, O, opr)*	*B*	*UR({MO&A}, O, opr)*
Mrg → Shr → Rev → Rep (B)	*UR({MO&A}, O, opr)*	*A*	*UR({MO&B}, O, opr)*
Mrg → Shr → Rep (C)	*UR({MO&A&B}, O, opr)*	*B*	*UR({MO&A&C}, O, opr)*
Mrg → Shr → Rep → Rev (C)	*UR({MO&A&C}, O, opr)*	*C*	*UR({MO&A}, O, opr)*
Revoke			

Rev (MO)	*UR({MO}, O, opr)*	*MO*	*UR({MO}, O, opr)*
Rev → Rep (A)	*UR({MO}, O, opr)*	*MO*	*UR({A}, O, opr)*
Rev → Rep → Shr (B)	*UR({A}, O, opr)*	*A*	*UR({A|B}, O, opr)*
Rev → Rep → Shr → Mrg (C)	*UR({A|B}, O, opr)*	*A,B*	*UR({A&B&C}, O, opr)*
Rev → Rep → Mrg (B)	*UR({A}, O, opr)*	*A*	*UR({A&B}, O, opr)*
Rev → Rep → Mrg → Shr (C)	*UR({A&B}, O, opr)*	*A,B*	*UR({A&B&C}, O, opr)*
Rev → Shr (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO|A}, O, opr)*
Rev → Shr → Mrg (B)	*UR({MO|A}, O, opr)*	*MO,A*	*UR({MO&A&B}, O, opr)*
Rev → Shr → Mrg → Rep (C)	*UR({MO&A&B}, O, opr)*	*B*	*UR({MO&A&C}, O, opr)*
Rev → Shr → Rep (B)	*UR({MO|A}, O, opr)*	*A*	*UR({MO|B}, O, opr)*
Rev → Shr → Rep → Mrg (C)	*UR({MO|B}, O, opr)*	*MO,B*	*UR({MO&B&C}, O, opr)*
Rev → Mrg (A)	*UR({MO}, O, opr)*	*MO*	*UR({MO&A}, O, opr)*
Rev → Mrg → Rep (B)	*UR({MO&A}, O, opr)*	*A*	*UR({MO&B}, O, opr)*
Rev → Mrg → Rep → Shr (C)	*UR({MO&B}, O, opr)*	*MO,B*	*UR({MO&B&C}, O, opr)*
Rev → Mrg → Shr (B)	*UR({MO&A}, O, opr)*	*MO,A*	*UR({MO&A&B}, O, opr)*
Rev → Mrg → Shr → Rep (C)	*UR({MO&A&B}, O, opr)*	*B*	*UR({MO&A&C}, O, opr)*

Table 7.1: Possible permutations for use-rights models

Table 7.3

Below is table 7.3, illustrating meta-rights allocation sequences, referred by section 7.1.2.

Meta-Rights			
Allocation Sequence	**Start State**	**Target**	**End State**
Replace			
Rep (A)	*MR({MO}, O, opr)*	*MO*	*MR({A}, O, opr)*
Rep → Shr (B)	*MR({A}, O, opr)*	*A*	*MR({A|B}, O, opr)*
Rep → Shr → Mrg (C)	*MR({A|B}, O, opr)*	*A,B*	*MR({A&B&C}, O, opr)*
Rep → Shr → Mrg → Rev (C)	*MR({A&B&C}, O, opr)*	*C*	*MR({A&B}, O, opr)*
Rep → Shr → Rev (B)	*MR({A|B}, O, opr)*	*B*	*MR({A}, O, opr)*
Rep → Shr → Rev → Mrg (B)	*MR({A}, O, opr)*	*A*	*MR({A&B}, O, opr)*
Rep → Mrg (B)	*MR({A}, O, opr)*	*A*	*MR({A&B}, O, opr)*
Rep → Mrg → Rev (B)	*MR({A&B}, O, opr)*	*B*	*MR({A}, O, opr)*
Rep → Mrg → Rev → Shr (B)	*MR({A}, O, opr)*	*A*	*MR({A|B}, O, opr)*
Rep → Mrg → Shr (C)	*MR({A&B}, O, opr)*	*A,B*	*MR({A&B&C}, O, opr)*
Rep → Mrg → Shr → Rev(C)	*MR({A&B&C}, O, opr)*	*C*	*MR({A&B}, O, opr)*
Rep → Rev (A)	*MR({A}, O, opr)*	*A*	*MR({A}, O, opr)*
Rep → Rev → Shr (B)	*MR({A}, O, opr)*	*A*	*MR({A|B}, O, opr)*

Rep → Rev → Shr → Mrg (C)	*MR({A|B}, O, opr)*	*A,B*	*MR({A&B&C}, O, opr)*
Rep → Rev → Mrg (B)	*MR({A}, O, opr)*	*A*	*MR({A&B}, O, opr)*
Rep → Rev → Mrg → Shr (C)	*MR({A&B}, O, opr)*	*A,B*	*MR({A&B&C}, O, opr)*
Share			
Shr (A)	*MR({MO}, O, opr)*	*MO*	*MR({MO|A}, O, opr)*
Shr → Mrg (B)	*MR({MO|A}, O, opr)*	*MO,A*	*MR({MO&A&B}, O, opr)*
Shr → Mrg → Rev (B)	*MR({MO&A&B}, O, opr)*	*B*	*MR({MO&A}, O, opr)*
Shr → Mrg → Rev → Rep (B)	*MR({MO&A}, O, opr)*	*A*	*MR({MO&B}, O, opr)*
Shr → Mrg → Rep (C)	*MR({MO&A&B}, O, opr)*	*B*	*MR({MO&A&C}, O, opr)*
Shr → Mrg → Rep → Rev (C)	*MR({MO&A&C}, O, opr)*	*C*	*MR({MO&A}, O, opr)*
Shr → Rev (A)	*MR({MO|A}, O, opr)*	*A*	*MR({MO}, O, opr)*
Shr → Rev → Rep (A)	*MR({MO}, O, opr)*	*MO*	*MR({A}, O, opr)*
Shr → Rev → Rep → Mrg (B)	*MR({A}, O, opr)*	*A*	*MR({A&B}, O, opr)*
Shr → Rev → Mrg (A)	*MR({MO}, O, opr)*	*MO*	*MR({MO&A}, O, opr)*
Shr → Rev → Mrg → Rep(A)	*MR({MO&A}, O, opr)*	*A*	*MR({MO&B}, O, opr)*
Shr → Rep (B)	*MR({MO|A}, O, opr)*	*A*	*MR({MO|B}, O, opr)*
Shr → Rep → Mrg (C)	*MR({MO|B}, O, opr)*	*MO,B*	*MR({MO&B&C}, O, opr)*
Shr → Rep → Mrg → Rev (C)	*MR({MO&B&C}, O, opr)*	*C*	*MR({MO&B}, O, opr)*
Shr → Rep → Rev (B)	*MR({MO|B}, O, opr)*	*B*	*MR({MO}, O, opr)*
Shr → Rep → Rev → Mrg (A)	*MR({MO}, O, opr)*	*MO*	*MR({MO&A}, O, opr)*
Merge			
Mrg (A)	*MR({MO}, O, opr)*	*MO*	*MR({MO&A}, O, opr)*
Mrg → Rev (A)	*MR({MO&A}, O, opr)*	*A*	*MR({MO}, O, opr)*
Mrg → Rev → Rep (A)	*MR({MO}, O, opr)*	*MO*	*MR({A}, O, opr)*
Mrg → Rev → Rep → Shr (B)	*MR({A}, O, opr)*	*A*	*MR({A|B}, O, opr)*
Mrg → Rev → Shr (A)	*MR({MO}, O, opr)*	*MO*	*MR({MO|A}, O, opr)*
Mrg → Rev → Shr → Rep (B)	*MR({MO|A}, O, opr)*	*A*	*MR({MO|B}, O, opr)*
Mrg → Rep (B)	*MR({MO&A}, O, opr)*	*A*	*MR({MO&B}, O, opr)*
Mrg → Rep → Shr (C)	*MR({MO&B}, O, opr)*	*MO,B*	*MR({MO&B&C}, O, opr)*
Mrg → Rep → Shr → Rev (C)	*MR({MO&B&C}, O, opr)*	*C*	*MR({MO&B}, O, opr)*
Mrg → Rep → Rev (B)	*MR({MO&B}, O, opr)*	*B*	*MR({MO}, O, opr)*
Mrg → Rep → Rev → Shr (A)	*MR({MO}, O, opr)*	*MO*	*MR({MO|A}, O, opr)*
Mrg → Shr (B)	*MR({MO&A}, O, opr)*	*MO,A*	*MR({MO&A&B}, O, opr)*
Mrg → Shr → Rev (B)	*MR({MO&A&B}, O, opr)*	*B*	*MR({MO&A}, O, opr)*
Mrg → Shr → Rev → Rep (B)	*MR({MO&A}, O, opr)*	*A*	*MR({MO&B}, O, opr)*
Mrg → Shr → Rep (C)	*MR({MO&A&B}, O, opr)*	*B*	*MR({MO&A&C}, O, opr)*
Mrg → Shr → Rep → Rev (C)	*MR({MO&A&C}, O, opr)*	*C*	*MR({MO&A}, O, opr)*

Revoke			
Rev (MO)	*MR({MO}, O, opr)*	*MO*	*MR({MO}, O, opr)*
Rev → Rep (A)	*MR({MO}, O, opr)*	*MO*	*MR({A}, O, opr)*
Rev → Rep → Shr (B)	*MR({A}, O, opr)*	*A*	*MR({A|B}, O, opr)*
Rev → Rep → Shr → Mrg (C)	*MR({A|B}, O, opr)*	*A,B*	*MR({A&B&C}, O, opr)*
Rev → Rep → Mrg (B)	*MR({A}, O, opr)*	*A*	*MR({A&B}, O, opr)*
Rev → Rep → Mrg → Shr (C)	*MR({A&B}, O, opr)*	*A,B*	*MR({MO&A&B}, O, opr)*
Rev → Shr (A)	*MR({MO}, O, opr)*	*MO*	*MR({MO|A}, O, opr)*
Rev → Shr → Mrg (B)	*MR({MO|A}, O, opr)*	*MO,A*	*MR({MO&A&B}, O, opr)*
Rev → Shr → Mrg → Rep (C)	*MR({MO&A&B}, O, opr)*	*B*	*MR({MO&A&C}, O, opr)*
Rev → Shr → Rep (B)	*MR({MO|A}, O, opr)*	*A*	*MR({MO|B}, O, opr)*
Rev → Shr → Rep → Mrg (C)	*MR({MO|B}, O, opr)*	*MO,B*	*MR({MO&B&C}, O, opr)*
Rev → Mrg (A)	*MR({MO}, O, opr)*	*MO*	*MR({MO&A}, O, opr)*
Rev → Mrg → Rep (B)	*MR({MO&A}, O, opr)*	*A*	*MR({MO&B}, O, opr)*
Rev → Mrg → Rep → Shr (C)	*MR({MO&B}, O, opr)*	*MO,B*	*MR({MO&B&C}, O, opr)*
Rev → Mrg → Shr (B)	*MR({MO&A}, O, opr)*	*MO,A*	*MR({MO&A&B}, O, opr)*
Rev → Mrg → Shr → Rep (C)	*MR({MO&A&B}, O, opr)*	*B*	*MR({MO&A&C}, O, opr)*

Table 7.3: Table depicting all possible cases for meta-rights models