ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 11(31), DOI: 10.17485/ijst/2018/v11i31/130889, August 2018*

Formal Validation of Erratic Delegation of Roles with UML and OCL

Saman Shahid Qureshi*, Mohsin Memon, Tehseen Hafiz and Pirah Memon

Department of Software Engineering, Mehran University of Engineering and Technology, Jamshoro, Sindh – 76062, Pakistan;
engr.samq@gmail.com, memohsin@gmail.com, tehseen_hafiz@hotmail.com, Pirahmemon01@gmail.com

Abstract

Objective: In emergency situations, when the responsible users for a task in the organization are not available or inadequate, the task should be delegated to eligible candidates satisfactorily. **Method**: In this paper, we proposed a mechanism for the specification and validation of constraints which can be applied during erratic delegation of roles. We have used Object Constraint Language (OCL) for specification of constraints. Here, UML Specification Environment (USE) tool is employed for the validation of the constraints that will be verified before and after the role delegation of the appropriate users. **Findings**: Focusing on the present condition of every single organization, there are a number of employees from all over the world who face many problems when distinct tasks are assigned to them. The entire scenario becomes more confused when a simple task is transferred to a user for the delegation, whose conditions need to be explicitly defined. Our delegation model helps to solve this. The results of the proposed delegation mechanism are promising since it is ensured that delegation occurs only when the pre and post conditions are satisfied. The user, who is being granted the role during an emergency, enjoys the same authority as being enjoyed by the actual user. The proposed model can be employed in a variety of situations for role/job delegation, when the organization requires a proper delegation mechanism for job assignment. **Novelty/Improvements:** In this paper, a validation mechanism has been proposed for the delegation of job roles to and from users with certain conditions and its validation has been carried. Here delegation process is being implemented over two different organizations and results are conferred.

Keywords: Erratic Environment, Formal Language, Specification, Validation, RBAC, USE Tool

1. Introduction

In any organization security is the most challenging task to handle. Problem arises in those organizations where numerous employees are working and the organizations have to assign a separate task to the individual users frequently. Therefore, it becomes complex to work out on different tasks assigned to various users. Researchers proposed a solution named as (RBAC) Role Based Access Control model[1]. In situations when number of clients requiring user's attention increased to certain limits or when a particular responsible user is unavailable on that current location and yet on other hand an emergency has been occurred, then role must be transferred to another person.

Therefore, role delegation and privileges seek by the certain constraints which must be snag and only then job assignment would be done or delegated to the user. For example: Asad wants to delegate task "VAT submission Review" to Saad in his absence on Monday Morning

*Author for correspondence

before board meeting starts or Professor Areej requires to delegate her role "Conduct Lecture" task to the other professor as she is not available in college. Researchers[2–4] developed the delegation models for assigning of delegated roles to the other user. In an emergency, erratic temporary delegation of rights of one user to others is necessary to carry out the work.

As the researcher in[5] highlights a mechanism that is completely bounded to the role delegation and revocation periodically. For that reason, it is imperative to develop a mechanism that can specify and validate constraints in the unpredictable/emergency situation in order to delegate roles so that work should not be stopped or delayed due to the absence of the responsible user, who is not available on location or busy in other chores. To monitor the job assignment, whether it is granted correctly or not it can be easily done by the help of formal language. The main purpose of this paper portrays a mechanism for role delegation with the support of pre and post conditions that are applied on the delegation method of two different organizations, where specification of constraints has been done in formal language and validation has been obtained by the assistance of use Tool (UML based specification) tool. Two scenarios are specially designed to serve this aspiration using Unified Modeling Language (UML) and Object constraint Language (OCL). The use tool is used to validate the delegation mechanism. In this paper section 2 describes the related technologies. Section 3 discusses the whole process of erratic environment based specification and validation of authorized delegates. Finally, section 4 consists of conclusion of this paper and an overview of future work.

2. Related Technologies

2.1 Role Based Access Control (RBAC)

RBAC was initially introduced by[1] which was later become an ANSI standard by NIST National Institute of Standards and Technologies in 2001. A part from RBAC, two other security models are also being used, named as, Discretionary Access Control (DAC) and the Mandatory Access Control (MAC). Currently they are the most ubiquitous security models that are frequently being used. Focusing specially towards the giant systems, RBAC assists to improve the entangled access control administrations. The four main components of this model are users, permissions, roles and sessions. The vital encouragement behind RBAC is the authorization.

2.2 Object Constraint Language (OCL)

The expressions, on object oriented models can be described by OCL. All these expressions are persistently defined pre and post conditions that must investigate over objects that are described in a model. In fact, UML and OCL are used in combination to create the models[6]. OCL is basically a formal language as well as a text based language. There upon, it enhances the understandability of models. OCL is not restricted to only UML models; it can be used by various other software tools, languages and methods for the constraints specification with in their models.

2.3 Use Tool

Figure 1 shows a software tool which allows the software modelers to specify UML models and validates OCL constraints by checking system and by execution of typical operational sequences[7]. It is an open source software tool. The specified constraints are formally investigated by the developers against their outlook and they can also drive formal method properties as a result with in their UML model.

The automated generation of non-trivial systems states, interactive monitoring of pre and post conditions as well as the invariants can be performed within this tool. The use API's execute Cmd ( ) method will be used by USE tool to facilitate the execution of state manipulation commands. Our UML models and OCL constraints can be specified in Use tool using a textual description. The information about the classes, their associations, invariants, and pre and post conditions of a current working model can be seen directly into the project browser that is present in use tool.

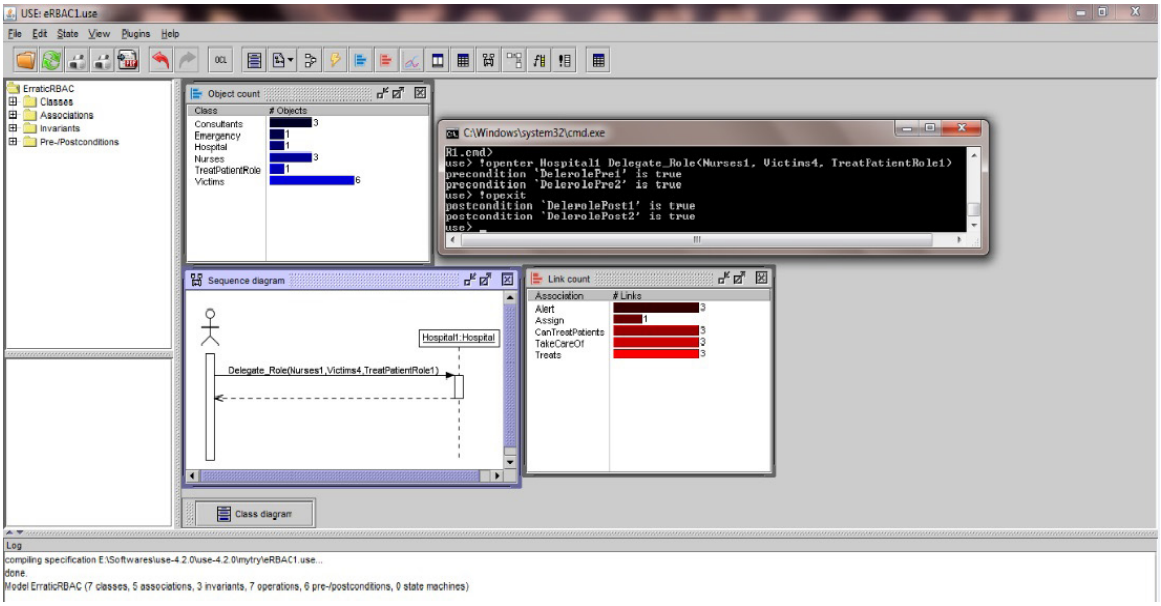

Figure 1. The use tool.

2.4 Delegation Model

The term delegation is a decentralized approach. This section edifies the introduction of the delegation model that has been given by[2]. In the correspondence to that a role can be delegated by a user if the following conditions are influenced.

a. The responsible person the (USER) has the all authority delegate his role.
b. Before role delegation there are certain restrictions that must be fulfilled by the delegator. In section 3 it has been discussed in scenario 1 and 2.
c. Delegator should make maximum number of validated delegations.

3. Specification and Validation of Delegation of Authority In USE

Within this section, the proposed solution has been discussed by the specification of two example scenarios in OCL. Firstly, delegation schemes are being formally specified that are discussed in section 2 (2.4) with OCL and later with the help of use tool, the erratic delegation mechanism is validated. The initial step is to check pre and post condition of delegation method which is called by delegator.

3.1 Example Scenario: 1

3.1.1 Specification of Delegation of Authority in OCL

Here in Figure 2 we elaborate the complete scenario 1 with the help of a diagram in which we discuss a formal approach for the specification of delegation scheme. Considering Figure 3 class diagram of example scenario 1 (Hospital emergency scenario), that consists of 6 major classes of consultants, hospital, nurses, victims, emergency and treat patient role and all are specified in use tool. These classes are used for the explanation of delegation mechanism.

The class hospital is used for creating objects for all consultants. The hospital class consists of one delegation method named as delegate_ role ( ), that will actually assign and delegate role assignment respectively. When emergency arrives, hospital alerts all their consultants immediately to counter this situation and treat all the victims as soon as possible. Facing such problematic situations like when the number of consultants is not enough or their presence is almost difficult within hospital; nurses

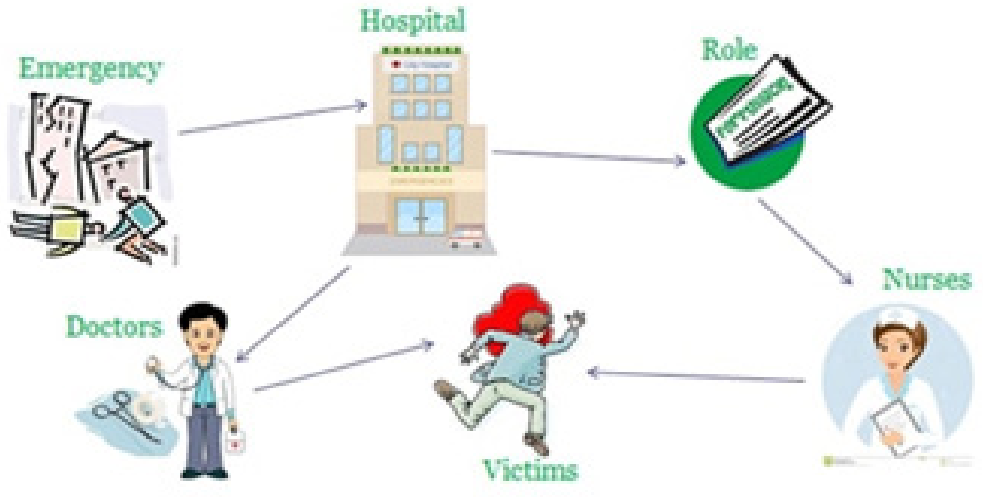

Figure 2. Hospital emergency scenario.

are also available for their help. However, the nurses cannot treat victims on their own. While considering such circumstances where emergency sources are natural disasters and the ratio of injured victims are infinite, the nurses should get the authority to treat the victims. The nurses will take the proper decisions without wasting

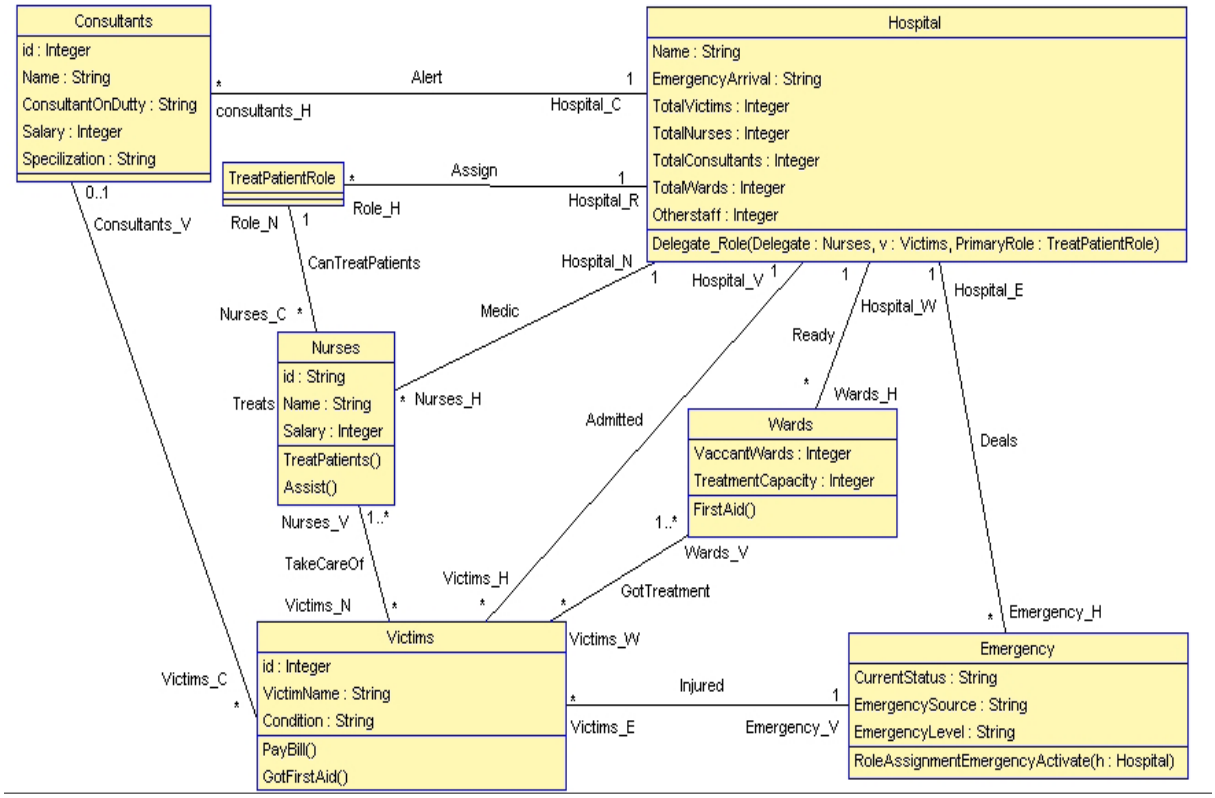

Figure 3. Class diagram of hospital emergency scenario.

```
Hospital::
   Delegate_Role(Delegate : Nurses, v : Victims,
                 Primary Role: TreatPatientRole)

  pre DelerolePre1: Delegate.isDefined()

  pre DelerolePre2:self.TotalConsultants <= self.TotalVictims

  postDelerolePost1:Delegate.Role_N -> includes(Primary Role)

  post DelerolePost2:Delegate.Victims_N -> includes(v)
```

Figure 4. Pre and Post Conditions of delegation method.

time and they will also be allowed to perform minor surgeries as well if it is needed for saving the precious lives of the unfortunate injured victims.

The constraints for the delegation model has been specified in object constraint language as pre and post conditions of the following method delegate_ role ( ) as shown in Figure 4 here hospital method delegate_ role () contains pre and post conditions: the pre-condition dele role pre1 ensures that the object of nurse class exists and the pre-condition dele role pre 2 guarantees that the total numbers of consultants are less than or equal to the number of victims. If both the pre-conditions are true then the use tool will satisfy the post conditions. The post condition post dele role post 1 verifies the role that has been actually delegated successfully to the nurse. The post condition dele role post 2 ensures that the nurse got the permission to treat the victims.

3.1.2 Validation of Delegation Constraints with Pre and Post Conditions

Initially class diagram of the hospital emergency scenario need to be loaded, which is explained in section 3 (3.1.1) it can be achieved with the help of use API's for loading this model which consists of the classes hospital, victims, consultants, emergency, nurses, treat patient role and it will use the method of delegation. Once a model will be loaded within use tool, the objects will be created by executing the state manipulation commands. The object diagram in Figure 5 hospital class has the information of the victims and has the authority to assign the treat patient role to the nurses at the time of an emergency.

According to Figure 5, a total number of 6 victims are admitted to the hospital and a total number of 3 consultants have already been called by the hospital for an emergency. According to the current situation, the number of consultants are less than the number of victims so it is the duty of the hospital to arrange the ample number of doctors or other relevant staff member who had enough experience, skill and knowledge to take a charge for the time being and deal with the emergency situation and treat the patients immediately. The hospital will assign task to the nurses to treat the victims. Nurses will get the authority for the certain time period. In order to assign treat patient role to the nurses, the method of the Hospital class delegate_ role ( ) is called which has the pre and post conditions, that are already specified within section 3 (3.1.1). For the role delegation to the nurses all these pre and post conditions must be satisfied. Figure 6 shows the model that has been validated before the delegation as well as after the delegation process of a role to

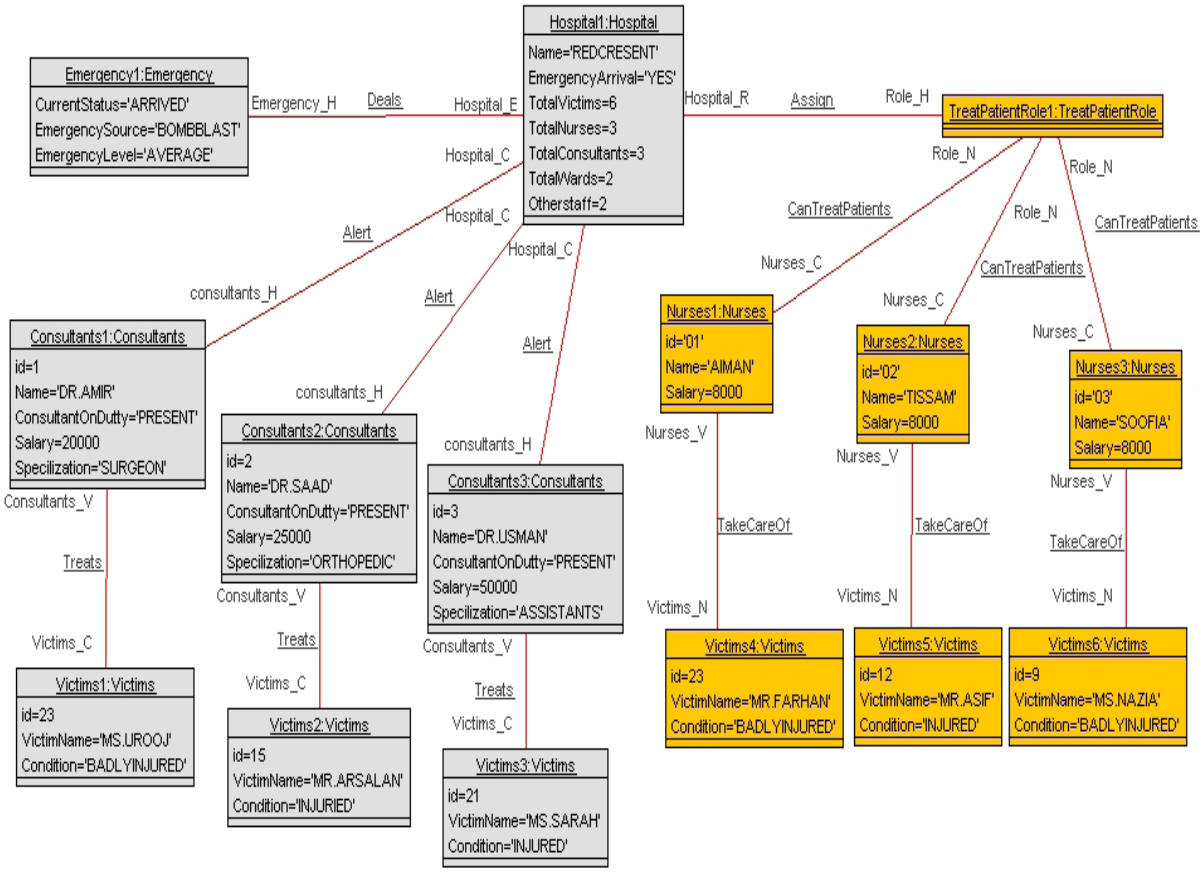

Figure 5. Object diagram of hospital emergency scenario.

```
R1.cnd)
use) !openter Hospital1 Delegate_Role(Nurses1, Victims4, TreatPatientRole1)
precondition 'DelerolePre1' is true
precondition 'DelerolePre2' is true
use) !opexit
postcondition 'DelerolePost1' is true
postcondition 'DelerolePost2' is true
use)
```

Figure 6. Validating pre and post conditions.

the Nurses. The delegation constraints using pre and post conditions in OCL are inscribed by use tool API.

This discussed validation mechanism assures that a role is delegated at the time of an emergency following by the conditions, that must be satisfied before delegation of a role to another user.

3.2 Example Scenario: 2

3.2.1 *Specification of Delegation of Authority in OCL*

Here in Figure 7 we elaborate the complete scenario 2 with the help of a diagram in which we discuss a formal approach for the specification of delegation scheme.

Scenario 2 (Figure 8) consists of five major classes university, department, chairman role, department affairs and professors are created. According to example scenario 2, all classes are associated with each other. Every

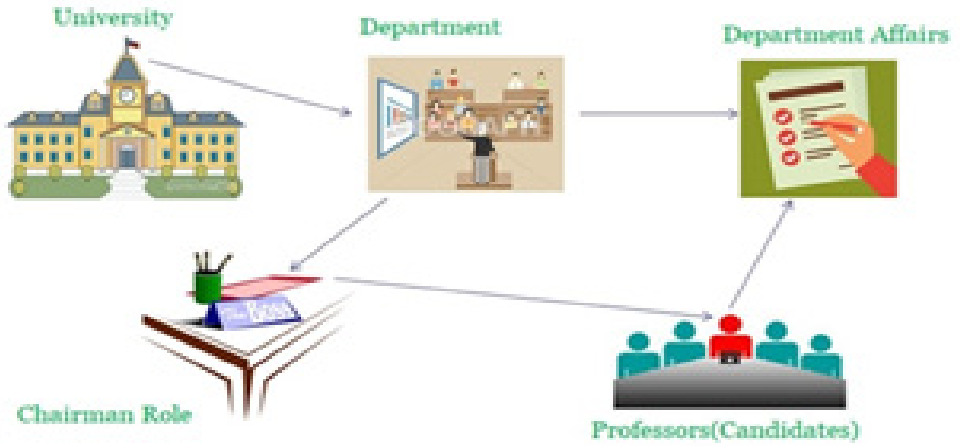

Figure 7. Department emergency scenario.

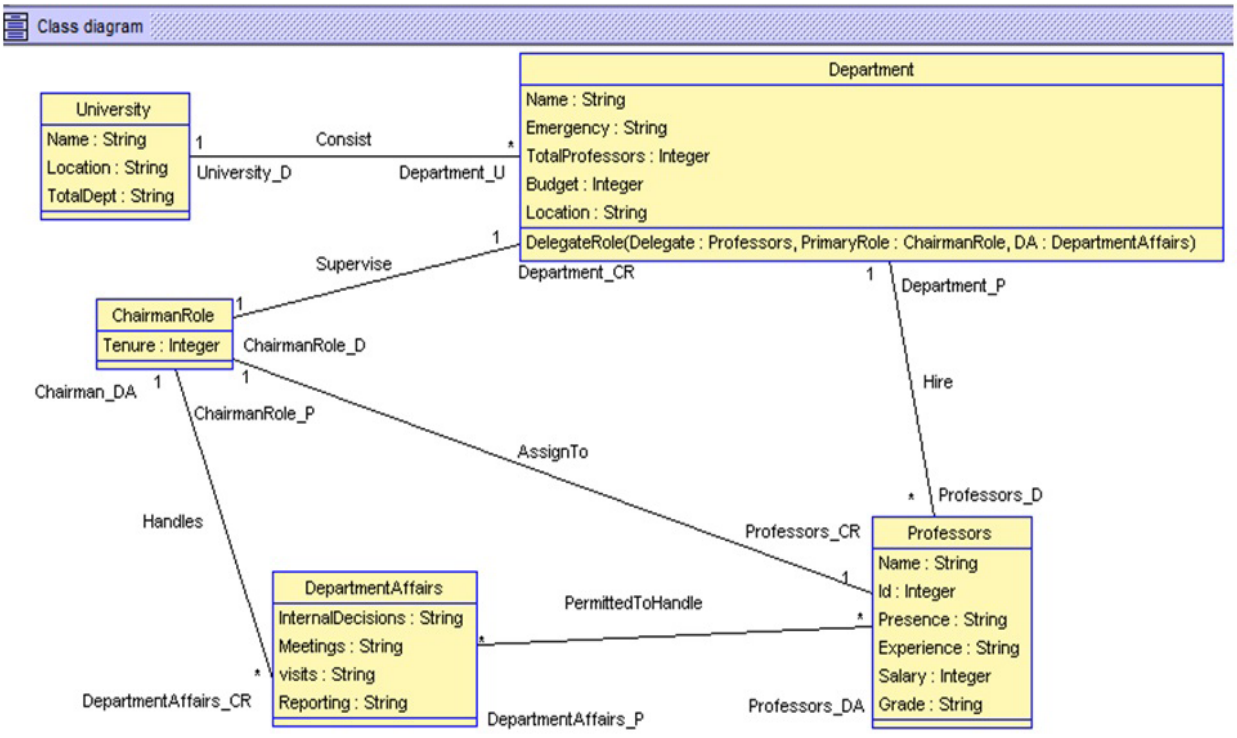

Figure 8. Class diagram of department emergency scenario.

```
Department::
        DelegateRole ( Delegate: Professors , PrimaryRole: ChairmanRole,
                       DA: DepartmentAffairs )

pre DelerolePre1 : Delegate.isDefined()

pre DelerolePre2 : self.Emergency='True'

pre DelerolePre3 : Delegate.Id=2

pre Delerolepre4 : Delegate.Presence='YES'


post DelerolePost1 : Delegate.ChairmanRole_P->includes(PrimaryRole)

post DelerolePost2 : Delegate.DepartmentAffairs_P->includes(DA)
```

Figure 9. Pre and post conditions of delegation.

University consists of more than one department and all those departments have more than one Professors working in it, each individual department also have a chairman to govern that department and will be taking care of all department matters. Focusing on the present situation, the current chairman is not available due to some reason such as vacation, so there should be a temporary eligible candidate (professor) to handle all the department affairs. Figure 9 shows the department class has a method named delegate role ( ) which takes the objects of class professors, chairman role and the department affairs as arguments for delegating, this role. The delegation of a role depends on these certain conditions. Firstly, the pre-condition dele role pre1 ensure that the object of professor class is defined and then following conditions are checked.

Pre-condition dele role pre 2 check and ensures the status to an emergency is set to true. The third pre condition checks whether the delegate to whom the role is being assigned is senior in the seniority list. The dele role pre 4 pre-condition verifies that the professor whose Id is 2, is present for completing the task. The dele role post 1 post condition verifies professor with Id=2 and the role is delegated successfully to him. Post condition dele role post 2 is to ensure that the relevant professor has got the permission to become a temporary chairman of the department and complete all remain task.

3.2.2 Validation of Delegation Constraints with Pre and Post Conditions

Exactly same process as scenario 1, initially class diagram of department emergency scenario need to be loaded which is explained in section 3(3.2.1). The model that consists of the following major class's university, department, department affairs, professors and chairman role. The object diagram in Figure 10 shows that a department has the information of their all faculty members presently working in to the department and each department has their own chairman to supervise and handle all department related responsibilities. Department class has an authority to assign the chairman role to the next senior professor who is next in line to become a future chairperson of that department. Figure 10 shows there are four professors, as an emergency arrives in the department then each of the professor's Ids and their availability will be checked accordingly for the selection of a deserving candidate. For example, professor with Id=1 is not available so likewise next candidate to be checked by its Id and availability till both of the conditions are satisfied. As it is shown that Professor with Id=2 satisfies all the pre-conditions.

Figure 11 the department emergency model has been validated, professor 2 as he/she satisfied all the conditions so the role has been successfully delegated. The validation

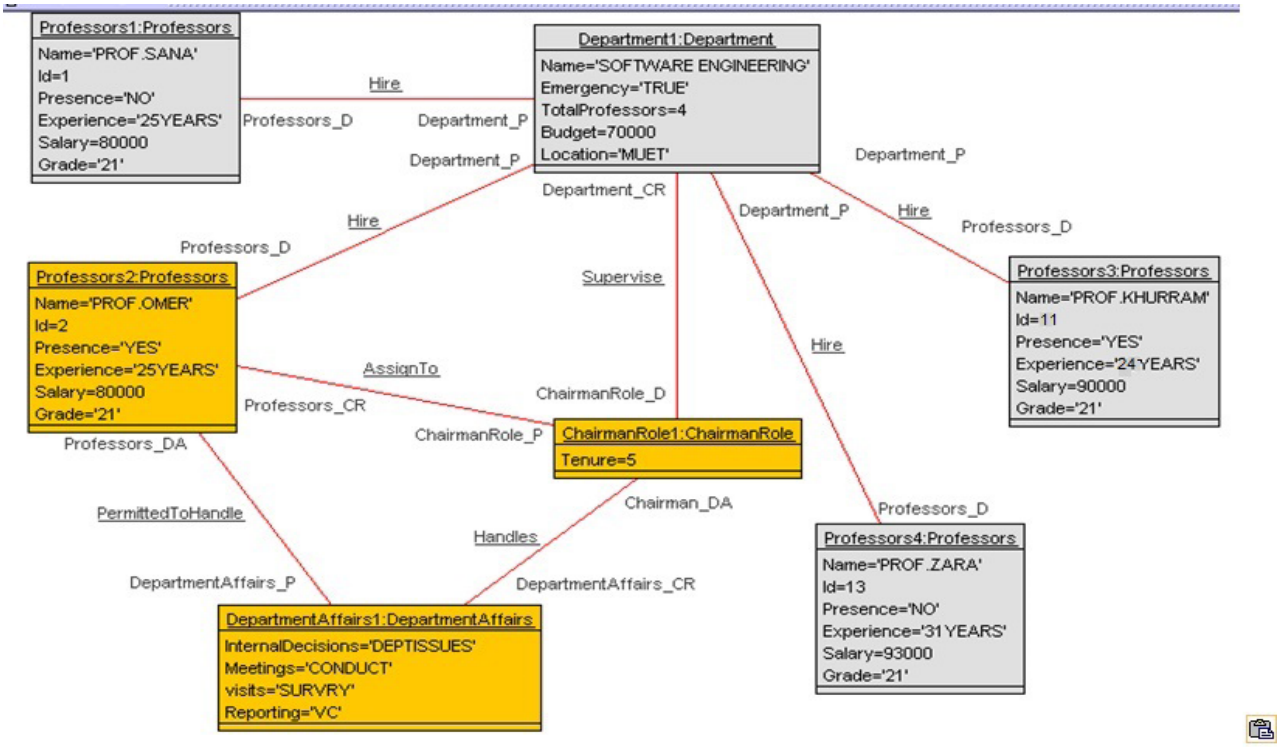

Figure 10. Object diagram of department emergency scenario.

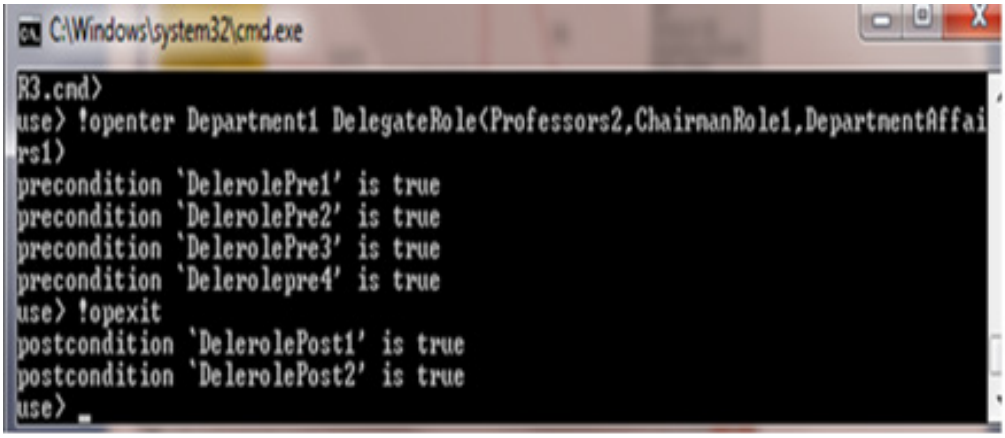

Figure 11. Validating pre and post condition.

mechanism discussed here assures that a role is delegated at the time of an emergency by following the constraints, which must be satisfied before delegating the role to another user.

Many researchers have given their tremendous efforts and significant work with n the domain of role delegation and revocation. They proposed various different models those highlights the user delegation of rights on the temporary basis such as[2,3,8–10]. In gave an idea of two level analyses for an enterprise especially for role based delegation and revocation policies[5] worked on the temporary role delegation and revocation that was limited for the

specific time duration. The key factor behind it was that role was delegated and revoked in a given time duration on temporary basis. The weakness of[9] is they introduce a permission based delegation model whose major drawback was that they gave a consolidated approach where the whole responsibility for the role delegation was of the security officer. In[10] discussed the delegation mechanism and focused on the temporal delegation of roles which deals with the fixed time period. In [11] their framework showcase various issues regarding delegation and revocation of roles.

4. Conclusion and Future Work

In this paper for better understanding of a job assignment, we have elaborated whole mechanism with the help of two examples scenarios. In which delegation process is being implemented on two different organizations. Our validation mechanism has been assisted in specifying and validating constraints in a formal language which deals with an erratic environment based delegation of rights of one user to other. The use tool API is used for the validating pre and post conditions of delegation methods. This work will help to understand the specification and validation of erratic delegation of roles with UML and OCL. By the help of this proposed mechanism the other delegation models may also be validated and use tool can also be further extended by creating a graphical user interface for erratic delegation of roles. This field of research still has a great potential of work to be explored in future.

5. Acknowledgement

The authors are highly grateful and like to express their gratitude to Prof. Dr. Mohammad Aslam Uqaili Vice Chancellor M.U.E.T, Jamshoro, Sindh, Pakistan for his support and motivation while conducting this research.

6. References

1. Ferraiolo DF, Sandhu R, Gavrila S, Kuhn DR, Chandramouli R. Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security (TISSEC). 2001; 4(3):224–74. https://doi.org/10.1145/501978.501980.
2. Zhang L, Ahn GJ, Chu BT. A rule-based framework for role-based delegation and revocation. ACM Transactions on Information and System Security (TISSEC). 2003; 6(3):404–41. https://doi.org/10.1145/937527.937530.
3. Nguyen TA, Su L, Inman G, Chadwick D. Flexible and manageable delegation of authority in RBAC. Advanced Information Networking and Applications Workshops, AINAW'07. 21st International Conference. 2007; 2:453–8. https://doi.org/10.1109/AINAW.2007.187.
4. Qiu W. Exploring user-to-role delegation in role-based access control. Eighth World Congress on the Management of eBusiness (WCMeB); 2007. p. 1–21. https://doi.org/10.1109/WCMEB.2007.47.
5. Memon MA, Hashmani M, Sohr K. Validation of temporary delegation and revocation of roles with uml and ocl. International Journal of Computer Theory and Engineering. 2010; 2(1):1–6. https://doi.org/10.7763/IJCTE.2010.V2.107.
6. Warmer JB, Kleppe AG. The object constraint language: getting your models ready for MDA. Addison-Wesley Professional. Computer; 2003. p. 1–206.
7. The USE tool: A UML-based specification environment [Internet]. [cited 2018 Jul 09]. Available from: https://sourceforge.net/projects/useocl/.
8. Sohr K, Drouineaud M, Ahn GJ, Gogolla M. Analyzing and managing role-based access control policies. IEEE Transactions on Knowledge and Data Engineering. 2008; 20(7):924–39. https://doi.org/10.1109/TKDE.2008.28.
9. Zhang X, Oh S, Sandhu R. PBDM: a flexible delegation model in RBAC. Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies; 2003. p. 149–57. https://doi.org/10.1145/775412.775431.
10. Wainer J, Kumar A. A fine-grained, controllable, user-to-user delegation method in RBAC. Proceedings of the tenth ACM symposium on Access control models and technologies; 2005. p. 59–66. https://doi.org/10.1145/1063979.1063991.
11. Barka E, Sandhu R. Framework for role-based delegation models. Computer Security Applications. ACSAC'00. 16th Annual Conference; 2000. p. 168–76. https://doi.org/10.1109/ACSAC.2000.898870.