DEVELOPING AN ABAC-BASED GRANT PROPOSAL WORKFLOW MANAGEMENT SYSTEM

by

Milson Munakami

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

December 2016

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Milson Munakami

Thesis Title: Developing an ABAC-based Grant Proposal Workflow Management System

Date of Final Oral Examination: 13th October 2016

The following individuals read and discussed the thesis submitted by student Milson Munakami, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Dianxiang Xu, Ph.D.	Chair, Supervisory Committee
Jyh-haw Yeh, Ph.D.	Member, Supervisory Committee
Jidong Xiao, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Dianxiang Xu, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

DEDICATION

This thesis is proudly dedicated to my beloved parents and lovely wife for their endless love, encouragement, and support.

ACKNOWLEDGEMENTS

I certainly did not get here by myself. Therefore, I want to thank all those who helped turned this dream into reality. I would like to express my sincere gratitude to all the people who supported me and made this thesis possible. First of all, I would like to extend my heartfelt gratitude to my wonderful advisor, Dr. Dianxiang Xu, who guided and walked me through this project. He is an excellent mentor and advisor. His invaluable advice, encouragement, motivation, and untiring guidance will help me in my future endeavor. I am also indebted to my committee members Dr. Jyh-haw Yeh and Dr. Jidong Xiao for their expert advice, help, and support.

I would like to extend my sincere appreciation to the Department of Computer Science at Boise State University for providing this excellent opportunity to pursue my Master of Computer Science degree and financially supporting me with a research assistantship without which the completion of my research would not have been possible. Besides, I would like to thank my dear friends and all the REU (Research Experiences for Undergraduates) students in Dr. Xu's lab for their lovely companionship. Otherwise, it would have been a mundane experience. My special, sincere and deep gratitude to my family members for their endless love, support, encouragement, motivation, and believing in me and my ability to succeed.

Lastly, I want to thank my wife, Nisha. During writing and completion of this thesis work, Nisha has supported me, encouraged me, and provided valuable suggestions to complete this paper on time.

ABSTRACT

In the advent of the digital transformation, online business processes need to be automated and modeled as workflows. A workflow typically involves a sequence of coordinated tasks and shared data that need to be secured and protected from unauthorized access. In other words, a workflow can be described simply as the movement of documents and activities through a business process among different users. Such connected flow of information among various users with different permission level offers many benefits along with new challenges. Cyber threats are becoming more sophisticated as skilled and motivated attackers, both insiders and outsiders, are equipped with advanced and diverse penetration tools and techniques. So apart from standard functional requirements, security is a critical requirement for such systems. We need to have a new approach to more secure design, configuration, implementation, and management of workflow systems. In this paper, we propose a new software design model when developing a workflow system that inherently decouples the system level functional requirements from the security specifications. This externalization of authorization from the code makes it more flexible to support dynamic business agility. Moreover, the proposed model is combined with contextual information to accommodate dynamic access control enforcement. The given architecture provides outstanding levels of control, security, privacy and compliance with regulatory standards by using more fine-grained static as well as dynamic Attribute Based Access Control (ABAC) policies. We also develop a viable implementation called Grant

Proposal Workflow Management System (GPWFMS) that supports not only functional and security specifications of workflow but also extended complex features like Obligations and Delegation of Authority which is lacking in the much existing literature.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

WFMS	**W**orkflow **M**anagement **S**ystem
GPWFMS	**G**rant **P**roposal **W**orkflow **M**anagement **S**ystem
ABAC	**A**ttribute **B**ased **A**ccess **C**ontrol
RBAC	**R**ole **B**ased **A**ccess **C**ontrol
XML	**E**xtensible **M**arkup **L**anguage
XACML	e**X**tensible **A**ccess **C**ontrol **M**arkup **L**anguage
DOA	**D**elegation **o**f **A**uthority
SOA	**S**ervice-**O**riented **A**rchitecture
OSP	**O**ffice of **S**ponsored **P**rograms
PI	**P**rincipal Investigator
Co-PI	**C**ollaborative **P**rincipal **I**nvestigator
IRB	**I**nstitutional **R**eview **B**oard
REST	**RE**presentational **S**tate **T**ransfer
API	**A**pplication **P**rogramming **I**nterface
JSON	**J**avaScript **O**bject **N**otation
PAP	**P**olicy **A**dministration **P**oint
PIP	**P**olicy **I**nformation **P**oint
PDP	**P**olicy **D**ecision **P**oint
PEP	**P**olicy **E**nforcement **P**oint

CHAPTER ONE: INTRODUCTION

With the advancement of cloud computing, Bring Your Own Device (BYOD) and Internet of Things (IoT), organizations are trying to adopt such new technologies to develop and implement autonomous workflow management systems (WFMSs). This digital transformation is bringing a new paradigm shift in the organization breaking the traditional approach of manual paper-based workflow management. Such online WFMSs focus on helping people to perform their tasks better and faster. However, the same level of security and automation is required by the organization along with promoting collaboration and information sharing among its stakeholders. As such fast-paced business processes are automated commonly referred as 'workflow automation' many security challenges need to be considered to streamline the work associated with each process step to make it more secure and flexible. Such dynamic and adaptive WFMS needs to provide a way to adapt to the vibrant and changing organizational needs to fulfill both system/functional and security requirements. As the threat landscape is changing and becoming more diverse and advanced, we need to architect, design, implement, and manage the security and privacy requirements in a way that allows users to focus on work and improve business operations rather than handling and tackling new security challenges associated with each task.

1.1 Background

Web-based WFMSs are widely becoming popular due to its high demand and adaptation of digital transformation in modern organizations. They are extensively used to

aid and streamline business processes in numerous application domains such as office automation, finance, and banking, healthcare, telecommunications, manufacturing, and production [1][2]. In such distributed workflow system, which usually deals with multiple users, shared resources, and environments; this is even more crucial to secure its critical assets. A general objective of such workflow management systems is to support increased workflow automation and security requirements in complex real-world environments involving heterogeneous, autonomous, and distributed information systems [3].

The increasing interest in replacing paper-based workflow into internet based online workflow systems make it vulnerable to security attacks and threats from outsiders as well as from insiders. Using autonomous workflow systems can leverage significant advantages to organizations by reducing paperwork, accelerating collaborations and providing better Quality of Service (QoS) to their customers. To fulfill and address such fundamental driving force behind each organization, developers need to have a firm understanding of their business objectives as well as security requirements. Apparently, due to developers' lack of understanding of business-oriented access control requirements, they can create many loopholes in the application. These security potholes can be easily exploited and impose high-security risks to the overall organizational goal.

In particular, the majority of available workflow systems do not yet support externalizing authorization from a business process. In these models, access is defined and controlled by each application's backend database or via hard-wiring within the code-level which can make them harder to address the dynamic organizational changes and restructuring processes. To make such a WFMS more secure and maintainable, we need to separate the business logic from the security features so that authorization logics do not

need to be within the code, but rather can be created and maintained external to the application.

With this separation of authorization from the functional business process, however, modeling, designing, composing and testing of such applications become harder and time-extensive. As it involves diverse and distributed stakeholders accessing the same resources from different environments and ‘context’ that is beyond the predefined organizational boundaries in such application. There is always risks of sensitive information disclosure/leakage, unauthorized data access/breaches, identity theft and lack of privacy protection. Workflow processes can be complex and deal with more sensitive data across many different users that require varying degrees of information confidentiality and data security mechanisms. Such workflow applications need to provide a way to control the access to the information based on user’s authority, privacy levels and other various implicit contexts.

Each workflow activities can act as an entry point for potential security threats and attacks, such as unauthorized access to the protected sensitive organizational information and leakage of critical personal data. The essential solution for data compliance and leakage prevention is controlling who can access what and when in accord to a set of pre-defined rules, routes, user roles, and privilege definitions. Such paradigm shift is increasing the complexity of workflow software architecture, design, and implementation. Hence, a more efficient and secure system design is needed to protect the significant flow of sensitive information from data theft and leakage.

1.1.1 Attribute Based Access Control

The main challenging problem of cyber security is the protection of shared data from unauthorized users for which different access control models are introduced. The concept of role-based access control (RBAC) began with early multi-user, and multi-application online systems pioneered in the 1970s [4]. The traditional RBAC model is insufficient in that it cannot describe fine-grained access constraints. It imposes many limitations for the granularity of permissions among distributed domains, resources, and users. It does not consider any other contextual information or object attributes except for role. From an enterprise perspective, RBAC is a passive access control model based on the direct assignment of roles and permissions that specify no time constraints, which can be exploited and can cause security threats. Such mechanism can be very messy and complicated if the organization has hundreds of thousands of users and similar roles that can lead to "role explosion". Changes to these associations between roles with privileges and users with roles are frequent and explicit. Manual change management is required and causes an unwanted delay on business processes. Such manual revocation of the users from assigned roles can cause big overhead for the organization administration. Also, inability to do manual revocation may result in many unforeseen security risks and may not correctly reflect the business requirements.

RBAC falls short of addressing dynamic fine-grained authorization at runtime. The shortcomings of traditional RBAC can be tackled by constructing a permission model using more fine-grained ABAC, which combines the flexible organization structure with the attribute based access control. ABAC is a relatively new paradigm for handling security policies. ABAC is more efficient logical access control methodology than RBAC where

authorization for activities is determined by analyzing attributes associated with the subject, object, action, and environment conditions. Due to its fine-grained nature, ABAC can be used to facilitate secure information sharing within the organization or federated environment. Unlike RBAC in which job function (role or identity) of a particular user defines an authority level, ABAC facilitates collaborative policy administration and auditing. ABAC explains not only WHO can access WHAT but also provides some additional context like WHEN, WHERE, WHY, and HOW. In simple words, ABAC relies upon the matching of attributes of the subject, attributes of the object, environment conditions, and their relationship with defined access control rules.

1.1.2 Case Study of a Workflow Management System

For this research work, we investigated ABAC model with the eXtensible Access Control Markup Language (XACML) Version 3.0 specification in a real-world application GPWFMS. In GPWFMS, we try to capture the real-world working process of University Grant Proposal Submission.

The regular activities in the proposal workflow life cycle are as shown in Figure 1.

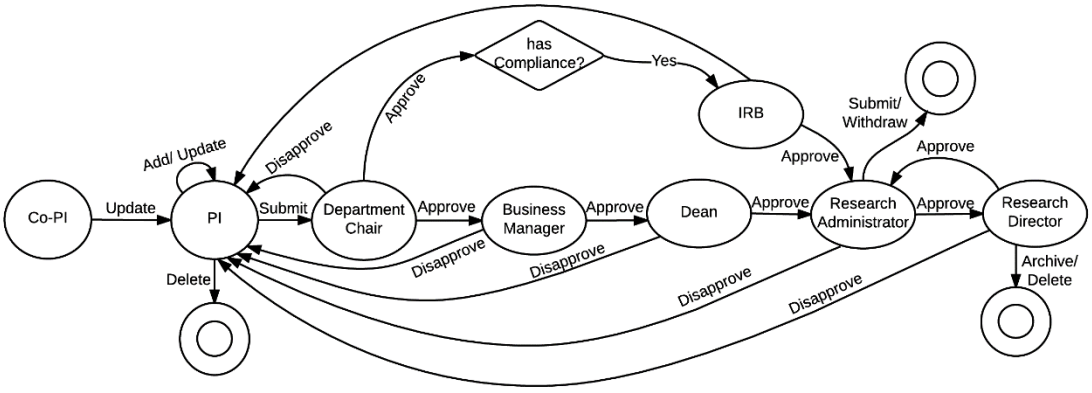


Figure 1 Proposal workflow life cycle without Delegation

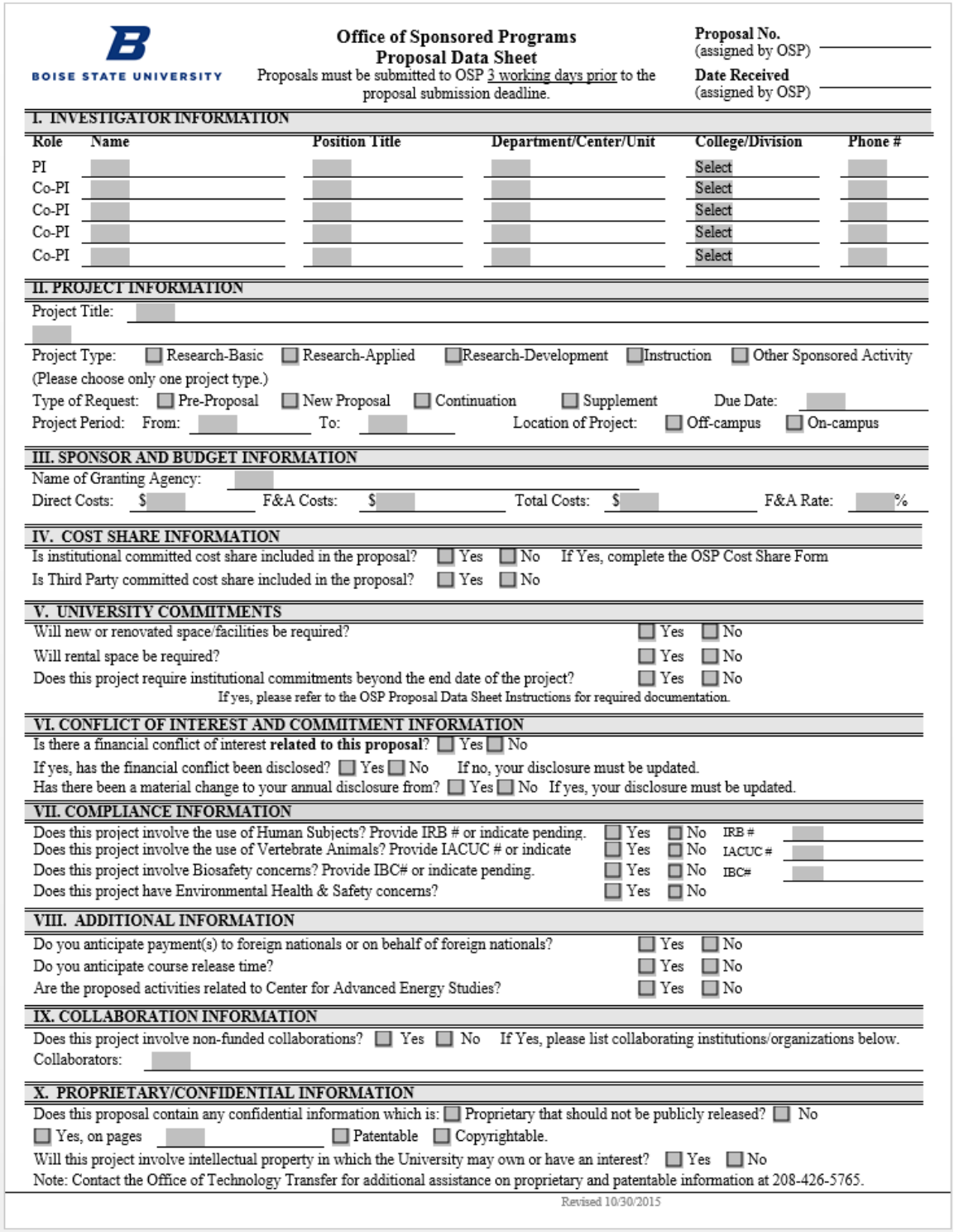
First of all, a research grant proposal is written and initiated by a *Principal Investigator (PI)* by filling the proposal information and relevant supporting documentation. It may include some *Collaborative Principal Investigators (Co-PI)* and *Senior Personnel* as co-authors or contributors. After getting the consent from each involved investigators, when the *PI* finds the proposal is ready to be submitted, he/she can submit it to the *Department Chair* for approval who will either return it or route it to the next phase in the workflow. After being approved by the chair, it will await for being reviewed by the *Business Manager*, the *Institutional Review Board (IRB)* and the *Dean*. This process can get even more complex and complicated if the proposal involves investigators from multiple departments, particularly for multidisciplinary efforts. In such case, all departments' authoritative personnel need to review and approve its content. Anyone of them can obstruct the overall proposal workflow process and can cause an unprecedented delay in completion of proposal submission. Once the proposal is approved by the *Dean* as well as reviewed by *IRB* if it involves any compliance issues to comply with Federal, State, and University regulations, then it must be routed to the *University Research Administrator* who can disapprove or withdraw it or can approve it by routing it to the *University Research Director*. Research Director can either refuse or delete the whole proposal or can give final approval for submission. Finally, once it gets approval from the *Research Director*, *University Research Administrator* can submit the proposal. Then *University Research Director* can archive the submitted proposal for future use.

As in the above-described usual scenario, it involves different activities that need administrative users with various position titles and privileges to engage and complete various tasks. Each activity within the workflow is associated with a subject who needs to

ensure the pending task is completed on time, and all obligations are fulfilled before and after any action is performed. We can view this complex workflow as a multi-layered state machine which needs to fulfill pre-conditions and post-conditions in each state and some specific event triggers it from one state to another.

In a typical paper-based proposal management workflow, an authorized user such as faculty needs to fill-up a lengthy data sheet paper form as shown in Figure 2 and Figure 3, with proposal information and hands it to the next level user such as Department Chair. Workflow tasks like approving/disapproving a proposal, budget reviewing, etc. which involves user authorization can be time-consuming. During each phase, the user's electronic signature plays a vital role as it indicates the consent from the user that corresponds to endorsement and commitment to the proposal. They can also request for revisions or additional information from the PI while reviewing the proposal. The most delaying factor usually is the length of time to reach a person and for that person to review the document. This task gets more complicated and tedious when the proposal involves other Co-PIs from different departments and need to be approved by authorized persons from each department. To convert such a tedious and time-consuming manual process into a flexible, reliable and more secure digital automated system is a challenge which respects the integrity of the workflow as shown in Figure 1 and Appendix B.

GPWFMS is a web-based workflow management system to automate and regulate the approval process of grant proposal submission which involves the creation, routing, and processing of grant proposals until completion. In particular, we are looking into a complicated setup of GPWFMS which may include various subjects trying to perform certain actions on shared resources that can alter data and control flow.



BOISE STATE UNIVERSITY

Office of Sponsored Programs
Proposal Data Sheet
Proposals must be submitted to OSP 3 working days prior to the proposal submission deadline.

Proposal No. (assigned by OSP) _____
Date Received (assigned by OSP) _____

I. INVESTIGATOR INFORMATION

Role	Name	Position Title	Department/Center/Unit	College/Division	Phone #
PI				Select	
Co-PI				Select	
Co-PI				Select	
Co-PI				Select	
Co-PI				Select	

II. PROJECT INFORMATION

Project Title:

Project Type: ☐ Research-Basic ☐ Research-Applied ☐ Research-Development ☐ Instruction ☐ Other Sponsored Activity
(Please choose only one project type.)
Type of Request: ☐ Pre-Proposal ☐ New Proposal ☐ Continuation ☐ Supplement Due Date:
Project Period: From: To: Location of Project: ☐ Off-campus ☐ On-campus

III. SPONSOR AND BUDGET INFORMATION

Name of Granting Agency:
Direct Costs: $ F&A Costs: $ Total Costs: $ F&A Rate: %

IV. COST SHARE INFORMATION

Is institutional committed cost share included in the proposal? ☐ Yes ☐ No If Yes, complete the OSP Cost Share Form
Is Third Party committed cost share included in the proposal? ☐ Yes ☐ No

V. UNIVERSITY COMMITMENTS

Will new or renovated space/facilities be required? ☐ Yes ☐ No
Will rental space be required? ☐ Yes ☐ No
Does this project require institutional commitments beyond the end date of the project? ☐ Yes ☐ No
If yes, please refer to the OSP Proposal Data Sheet Instructions for required documentation.

VI. CONFLICT OF INTEREST AND COMMITMENT INFORMATION

Is there a financial conflict of interest **related to this proposal**? ☐ Yes ☐ No
If yes, has the financial conflict been disclosed? ☐ Yes ☐ No If no, your disclosure must be updated.
Has there been a material change to your annual disclosure from? ☐ Yes ☐ No If yes, your disclosure must be updated.

VII. COMPLIANCE INFORMATION

Does this project involve the use of Human Subjects? Provide IRB # or indicate pending. ☐ Yes ☐ No IRB #
Does this project involve the use of Vertebrate Animals? Provide IACUC # or indicate ☐ Yes ☐ No IACUC #
Does this project involve Biosafety concerns? Provide IBC# or indicate pending. ☐ Yes ☐ No IBC#
Does this project have Environmental Health & Safety concerns? ☐ Yes ☐ No

VIII. ADDITIONAL INFORMATION

Do you anticipate payment(s) to foreign nationals or on behalf of foreign nationals? ☐ Yes ☐ No
Do you anticipate course release time? ☐ Yes ☐ No
Are the proposed activities related to Center for Advanced Energy Studies? ☐ Yes ☐ No

IX. COLLABORATION INFORMATION

Does this project involve non-funded collaborations? ☐ Yes ☐ No If Yes, please list collaborating institutions/organizations below.
Collaborators:

X. PROPRIETARY/CONFIDENTIAL INFORMATION

Does this proposal contain any confidential information which is: ☐ Proprietary that should not be publicly released? ☐ No
☐ Yes, on pages ☐ Patentable ☐ Copyrightable.
Will this project involve intellectual property in which the University may own or have an interest? ☐ Yes ☐ No
Note: Contact the Office of Technology Transfer for additional assistance on proprietary and patentable information at 208-426-5765.

Revised 10/30/2015

Figure 2 Grant Proposal Data Sheet Page 1

XI. CERTIFICATION/SIGNATURES

Investigators, department chairs, directors, deans certify that 1) the proposed activities are appropriate to the research, instruction, and public service mission of the University; 2) if funded all necessary resources as proposed will be provided for the project (i.e., cost share, personnel, facilities), and project expenditures that exceed the sponsor's award and/or payment upon completion of the project will be charged to the departmental account that you will identify at the time of award setup.

Principal or Co-Principal Investigators certify that 1) the information submitted within the application is true, complete and accurate to the best of the Investigator's knowledge; 2) all necessary resources to successfully complete the proposed project have been identified in the proposal; 3) the application is true, complete, and accurate to the best of my knowledge, 4) any false, fictitious, or fraudulent statements or claims may subject the PI to criminal, civil, or administrative penalties, 5) the PI agrees to accept responsibility for the scientific and programmatic conduct and financial oversight of the project and to provide the required progress reports; and 6) the PI shall use all reasonable and best efforts to comply with the terms, conditions, and policies of both the sponsor and the University. PIs should refer to http://web1.boisestate.edu/research/osp/standard-compliance.shtml for a list of responsibilities.

Department chairs and deans acknowledge that Facilities & Administrative costs for projects involving more than one college will be distributed in accordance with University policy 6100 unless otherwise directed in writing with approval from all deans involved.

Principal/Co-Investigator(s)	Date	Dept Chair(s) or Director(s)	Date
		Dean(s)	Date

Business Manager (if applicable) has reviewed this proposal. Initials: _____

Office of Sponsored Programs Administrative Use Only:

Flow-Through, List Agency: _____

Funding Source: ☐ Federal ☐ Federal Flow-Through ☐ State of Idaho Entity ☐ Private for Profit
☐ Non-Profit Organization ☐ Non-Idaho State Entity ☐ College/University ☐ Local Entity
☐ Non-Idaho Local Entity ☐ Tribal Government ☐ Foreign

CFDA No.: _____ Program No: _____
Program/Solicitation Title: _____

Recovery: ☐ Full Recovery ☐ No Recovery-Normal Sponsor Policy ☐ No Recovery-Institutional Waiver
☐ Limited Recovery-Normal Sponsor Policy ☐ Limited Recovery-Institutional Waiver
Base: ☐ MTDC ☐ TDC ☐ TC ☐ Other ☐ N/A

Is PI salary included in the proposal? ☐ Yes ☐ No If No, provide a Department ID for 1% minimum
PI Salary: _____ PI Fringe: _____ Department ID: _____
Institutional Cost Share Documented ☐ Yes ☐ No ☐ NA Third Party Cost Share Documented ☐ Yes ☐ No ☐ NA

Are subrecipients (subcontracts/subawards) anticipated? ☐ Yes ☐ No
Names of subrecipients: _____

PI Eligibility Waiver on File ☐ Yes ☐ No ☐ NA ☐ This Proposal Only ☐ Blanket
Conflict of Interest Forms on File ☐ Yes ☐ No ☐ NA Excluded party list has been checked ☐ Yes ☐ No ☐ NA

Proposal Notes: Research Administrator: ☐ DF ☐ LG ☐ LN

Send original to Office of Sponsored Programs, MS 1135 or osp@boisestate.edu.
Please send email to osp@boisestate.edu. to request a final copy of the Proposal Data Sheet.

Revised 10/30/2015

Figure 3 Grant Proposal Data Sheet Page 2

In GPWFMS, we automate this entire workflow lifecycle so that it is completely electronic and paperless. The new automated process saves time waiting for paperwork to traverse around the campus. It also provides a secure and central location to store and manage all relevant documentations. Organizations intended to enforce privacy and security regulations will have their access control policies and business rules based on functional and security requirements. The functional and security mechanisms such as privacy, access control, and usage control are defined and documented. These access level rules determine how proposal-related information is managed, processed, routed, and tracked to make decisions in every step. For example, one rule might be to have conditional routing of data and tasks based on the status of the proposal and user's context.

The Unified Modeling Language (UML) based Use Case diagrams with their textual descriptions are used to formulate such requirements. These formal specifications are translated into eXtensible Access Control Markup Language (XACML) based authorization policies by utilizing fine-grained ABAC model. Thus, it requires verification and validation of the correct access to the requested resources using subject's access levels which are determined by subject, action and resource's attributes. Attributes may be considered characteristics of entities that may be predefined and pre-assigned a value by an authority.

1.1.3 Obligation and Delegation of Authority

Along with making it more automated and secure, we need to consider the possibility of having many 'disconnected users' who can obstruct the flow of the task. 'Break-the-glass' is one approach which helps to prevent such workflow stagnation based on flexible and dynamic policies. In such break the glass scenarios, sophisticated features

such as system and user-level Obligations, Advice, Delegation of Authority (DOA), and Delegation of Obligations can be helpful so that the task can be completed on time.

Obligations are requirements that have to be fulfilled by the subject before (pre) or after (post) performing an action on a particular resource. For example, a pre-obligation requirement is that a user must sign the proposal with the current date time, initial and note before approving or disapproving it while it is waiting for his approval. As this need is to be fulfilled by a user, this is an example of user-level obligation. On the other hand, post-obligation is to notify all associated persons of that proposal about the change via email. The system performs such post-obligation as a system-level obligation. Moreover, in current existing workflow systems, there is no way we can impose obligations on any users based on policy rules.

Proposal workflow life cycle with complex delegation scenario is shown in Appendix B. Interestingly, issues of DOA can cause a critical security threat to the business as it provides more administrative authority to any new user (delegatee) in absence or consent of authority (delegator). Also, each delegation policy can have its obligation constraints known as delegation of obligations, which need to be enforced and fulfilled by the delegatee and the system.

1.2 Problem Statement

According to Workflow Management Coalition (WfMC), Workflow is defined as, "*The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules*" [5]. The WfMC has published a standardized security workflow model describing some security services that includes authentication, authorization,

access/usage control, audit, data privacy, data integrity and non-repudiation. Such standards clearly emphasize the major security objective of any workflow system is to prevent the unauthorized access of classified information. Overall, the workflow modeling lacks research and standardization on how to design and implement a reliable and secure workflow specification.

In the study of workflow secure access control models, the task-based access control (TBAC) and role-based access control (RBAC) are most commonly considered and applied [6]. As WFMSs are used for critical and strategic applications, security is an essential and fundamental part of such systems. Many Web-based workflow applications enhance their safety via access control systems [7][8][9]. Our goal of this research work is to improve the existing secure software design model that mainly advocates for the use of TBAC, RBAC [6] and ABAC without the concept of DOA and Obligations. The primary focus of the security in such model is based on their role in the organization which can quickly restructure or change in dynamic enterprises; which means the client codes need to be reconfigured and modified. NIST [10] indicates ABAC as a recommended access control model for promoting information sharing among diverse and disparate organizations.

Even though we are experiencing an unprecedented rise in the popularity of WFMSs, little has been done to take into account the standardization of access control constraints such as Separation of Duties (SoD), DOA and Obligations. Today's workflow systems need to provide the automation of a business process using more coordinated and collaborated execution of multiple tasks from different entities that may reside outside the inter-organizational boundaries at distributed environments. On the one hand, such intra-

boundaries access demands the system to support for continuous and collaborative business process that puts the business flows immediately and directly under the control of the people using the system. On the other hand, it needs to govern all security access control constraints via centralized and unified XACML policies.

The complexity of real-world workflow application requirements both functional and non-functional is revealing the limitations of the current security model design. The dominant traditional security access models are more discretionary and do not consider contextual information such as date, time, location and environments that allow intruders to bypass any defined security mechanisms easily. Existing state-of-art digital workflow solutions have security access controls hard coded at the application level, and also they do not specify complex access control constraints such as DOA and Obligations in policy level. Code-level access control logic making such systems rigid, incomplete, less secure and easy target to the security threats. When access decisions are embedded within the client applications, it makes it tough to update the decision criteria when the governing business rule changes. With such rigid software design patterns, it makes it harder to adapt any changes with the existing applications. Thus, there is a great need for flexibility in software design and implementation that supports dynamic changing of security policies based on DOA and obligation constraints. Improper design and implementation of such access control security constraints may increase critical complications.

Additionally, the presentation layer is all based on developer understanding of the domain. On the other hand, if we can leverage the power of XACML policy, we can implement the policy rules on presentation tier that can provide more personalized and

business rules oriented user interfaces. Moreover, such interfaces can reflect the changing need of operations in future without the need of re-coding the application.

In particular, we need to investigate various security concerns in a complex environment of GPWFMS. One of the many outstanding technical challenges of adaptive WFMS is that it needs to unify people and resources with diverse features into a more cohesive way. A secure online workflow system needs to comply with all security requirements of the organizations alongside their system objectives and should safeguard all the sensitive information at any point of time. We can achieve this by integrating organizational access control policies throughout the workflow activities. However, this does not mean that it needs to imply many restrictive measures during each action from the user to make it more secure and robust. Such restrictions may degrade the usability or user acceptance of the overall system and also can impact the system's performance.

1.3 Objective

Our main contribution is to propose and develop a more secure and reliable software design model that uses ABAC using XACML policy. These unified policies are driven by administrative delegation and access control with obligations rules which are flexible enough to manage and adapt complex system requirements. Using the latest specification of XACML profile, we can implement policy-driven interface design. Such policy-based capabilities demonstrate how we can use ABAC in presentation layer not just as a middle layer between service and database. This flexibility makes the system more configurable based on a comprehensive and formal set of governance rules rather than hard coded by a developer and provides a more personalized user experience.

These extensions in XACML standard are very helpful toward achieving sophisticated security features. However, it does not specify the kind of software design required to handle them properly. Such immaturity of XACML is making these new access control concepts less applicable and hence there are limited examples and implementations available. To the best of our knowledge, very few related work has been carried out in the real use case and implementation of such security model. Thus developed workflow management system can demonstrate a good use case for implementation of our proposed software design model that is simple to use and to administrate.

This challenge allows us to develop a good software architecture that can support system requirements which are common in the real-world dynamic organization. To fulfill such on-demand security requirement and replace the existing limitation of available solutions, we are proposing a new software design architecture which implements ABAC along with advanced access control concepts such as DOA and Obligation to model much closer to realistic business authorization scenarios. Also, this software model can externalize authorization by separating Database and Web Services access functionalities from business policies making it truly agile, powerful, and dynamic. The proposed software design and architecture makes the authorization mechanism more flexible and useful which simplifies the task complexity of security administrator and developers. The security administrator needs to write and update the XACML policies that cover all the functional and access control security requirements in a central repository. On the other hand, this approach helps developers focus on business-oriented problems rather than basic service implementations. As ABAC based policy rules do not require the creation or maintenance of hierarchical structure as in an RBAC model, such rules need less maintenance and

overhead. This model combines the advantages of the new fine-grained ABAC model along with other security access control constraints. Such combination reduces the risks of data breaches, sensitive data leakage and identity theft in an organization.

1.4 Outline

The paper is structured as follows. Chapter 1 briefly describes the challenges that exist in current software development practice, which illustrates the need for flexible and secure software architecture. Chapter 2 gives an overview of related work in this area. In Chapter 3, we outline all system requirements that the application needs to comply with and support. These requirements are the desirable criteria to evaluate the system design and implementation. Also, it explains how XACML access control policy can be used to express such security assertions rather than embedding them in code-level. Chapter 4 discusses the development and implementation of an authorization architecture enforcing our approach to support sophisticated features and requirements of the workflow system. Chapter 5 describes how the system requirements are used for evaluating our secure design, along with the result of our automated tests. Also, we explain some of the assumptions based on which our system security model is constructed. Chapter 6 summarizes our conclusions, together with the future direction for our research work. The paper also contains an appendix reporting the detail system use-cases textual description, functional and security requirements, test results and some policy rule specifications, XACML based request and response protocol format used by our proposed model.

CHAPTER TWO: RELATED WORKS

2.1 Access Control

To accomplish security needs of any adaptive workflows, we can implement access control mechanisms [7][8][9]. According to National Institute of Standards and Technology (NIST) - "*An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions.*" [10]. Access control is always necessary for organizations to offer proper data security and protection. In recent years, many secure access control models [8][11][12][13] are proposed and studied for collaborative and intra-organizational environments that express complicated access control constraint using traditional security methods. Unfortunately, those static access control models radically fail to meet new regulatory standards and safeguard compliance demand of a dynamic organization. In a workflow, security involves the implementation of a secure access control mechanisms to ensure that no subjects are allowed to perform unauthorized activities on given resources. However, the biggest problem is such objects can have dynamic attributes and characteristics based on the contextual information surrounding a request. Contemporary information security mechanisms are often immature or insufficient in addressing such demanding compliances due to lack of standardization.

In distributed systems, models and languages have been widely investigated to specify access and management of control policies [14]. With the advent of web services based Service-oriented Architectures (SOA), these frameworks are enhanced to meet security of the distributed environment. A typical approach is to assign users to one or more roles, and then to grant security to those roles known as RBAC [4][6]. The system should be able to define access control to those roles at several levels. Unfortunately, these information security mechanisms are insufficient to address the complex security requirements that are more fine-grained and need to support different collaborative activities such as pre/post obligations and delegation of tasks.

Attribute-based access control is proposed as the perfect access model to overcome the shortcoming of traditional RBAC model. Movahednejad et al. [15] describe comparative evaluation and taxonomy of state-of-the-art approaches. Similarly, in other papers [16][17][18], authors have described the advantages and benefits of ABAC model. By contrast, our model makes it more fine-grained access control by supporting the contextual information i.e. time, location and environmental state for any user requests.

2.2 XACML

Use of XACML-based expressive access control policies is proposed to protect the access of resources in distributed systems that facilitates dynamic access control [19]. Herrmann [20] also explains about the design of a conceptual and logical evaluation context model based on XACML 3.0 specifications.

XACML is XML-based declarative policy language for defining access control policies and a related processing model which permits the specification of authorizations as rules. Granular level of access control can be achieved in XACML as a specialized

implementation of ABAC. Furthermore, XACML is a generic framework recognized by OASIS standard[1] for access control which ideally provides standardization, expressiveness, modularity, interoperability, and efficiency [21][22]. The XACML specification defines a declarative fine-grained, attribute-based access control policy language, a reference architecture, and a processing model describing how to match access requests according to the stored policy rules. XACML standards address and determine how security authorization requests are handled internally.

An XACML policy P can be formalized using 5-tuple (S, R, A, C, O_b) [23], where S is a set of subjects, R is a set of resources, A is a set of actions, C is a set of permission conditions which can be evaluated to either true or false and O_b is a set of obligations. XACML architecture is a suitable choice for our model because of its:

i. Expressive power in expressing policies.
ii. Computational simplicity in access algorithms.
iii. A natural language translation from business policies to access rules.
iv. Standardized processing model which supports the externalization of the access decision from the business logic.

As shown in Figure 4, XACML Policy Language Model composes of many components. The policies may consist of different access control constraints in the form of policy sets, policies, decision rules, conditions, etc. for defining access level to the resources for a user.

The main elements of the XACML Policy Language model are:

1. Policy Sets: A policy set consists of one or more policies, other policy sets and a declaration for policy-combining algorithms.

[1] https://www.oasis-open.org/standards

2. Policies: A policy includes a set of rules, a resolution for appropriate rule-combining algorithms, a set of obligations and advice, and a target.

3. Rules: A Rule is the simplest unit of policy. A policy can comprise of one or many rules that can evaluate to Permit, Deny, Indeterminate, or Not Applicable.

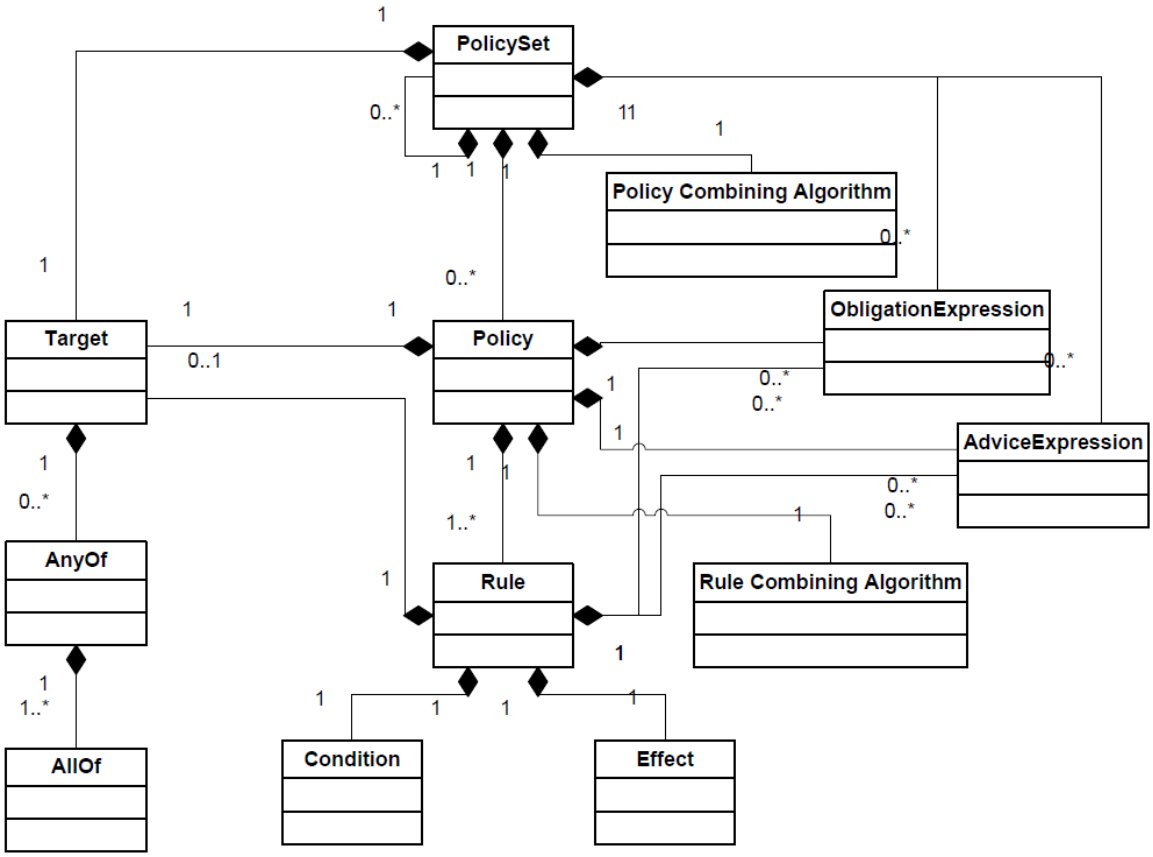


Figure 4 XACML Policy Language Model

Each access control rule may consist of a condition, an effect, and a target to provide the fine-grained security.

- Conditions are statements about attributes that can evaluate either True, False or Indeterminate.
- The effect returns value Permit or Deny based on the satisfied rule.

- Target in policy helps in determining whether or not a rule is relevant for a request.
- As a policy can have multiple rules, it is evident that it can generate different decisions based on different conflicting rules. To minimize that risk Rule-combining algorithms are used which resolve such conflicts and always try to outcome only one decision per policy.

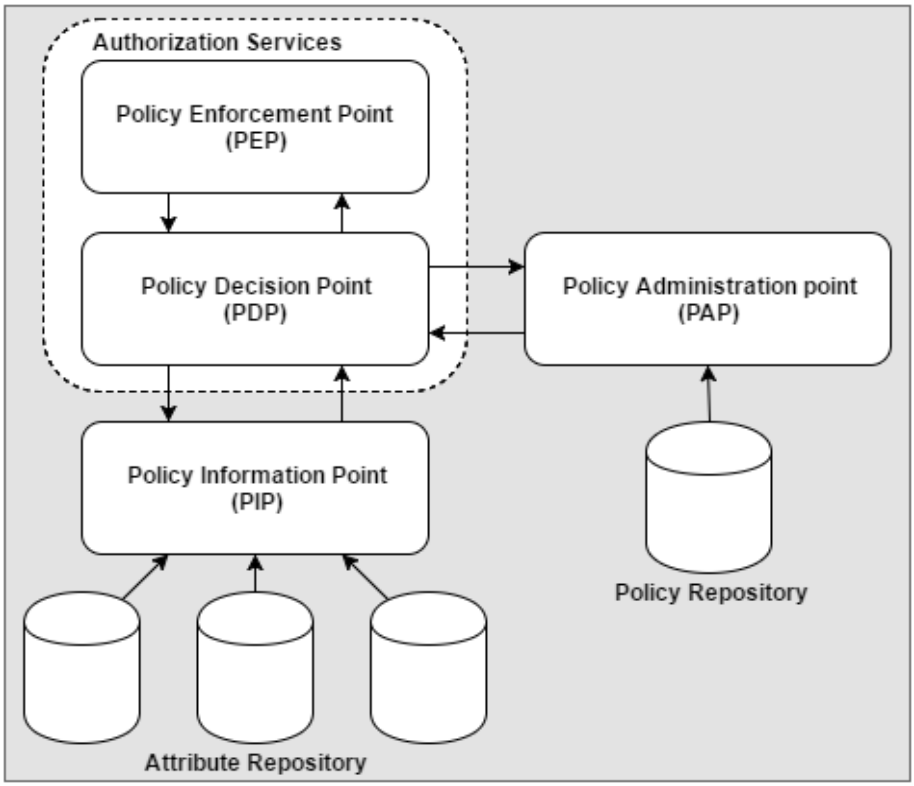


Figure 5 High-Level Design of XACML Enforcement Architecture

The XACML reference architecture as shown in Figure 5, highlights all the logical components of XACML as well as their internal interactions and authorization flows. It can be viewed as interactions of four top-level components as described below:

- The policy administrator defines and manages policies and policy sets at the Policy Administration Point (PAP). XACML supports a variety of

underlying infrastructures for policy and attribute storage. The policy repository stores the rules, policies, and policy sets that are used for access control.

- Policy Information Point (PIP) behaves as a metadata of attribute values (i.e. a resource, subject, environment conditions) and can be federated.
- The Policy Decision Point (PDP) analyzes the resource access request with the matching rules, policies, policy sets and returns a decision to the caller.
- The Policy Enforcement Point (PEP) forwards the incoming request for access or authorization decision to the XACML context handler with a predefined format that specifies the details about the attributes of the subjects, resources, actions, and the environment. Placement of PEP directly influences the overall system performance.
- Once the policy is evaluated successfully, the PEP will either permit access or deny the access to the service requester for the requested resource and action. Also, the decision includes associated obligations and advice along with the reply if any.

2.3 Obligation

Mbanaso et al. [24] proposed a model that uses obligations of trust to negotiate between the client and service provider to adequately preserve the user's privacy. This communication is based on XACML standard and applicable to be integrated into distributed access control systems. In the distributed settings, without more secure access control methods there is always the risk of leakage of business-critical and personal assets. Another paper by Sans et al. [25] explains how policy language can be used to express both

contextual permissions and obligations. Such usage control mechanisms are used while evaluating and enforcing the policies. Although, they lack the concept and support for dynamic changes in policies that are inevitable in today's distributed systems. Elrakaiby et al. [26] had formalized the enforcement and management of obligation policies in which they had used the concepts of action specification languages and the Event Condition Action based on different states of obligations. Unlike general two types of obligations i.e. *Pre* and *Post*, in this article, they also identified *Ongoing* obligations which are activated when resource usage starts. Also during the enforcement and fulfillment of usage-control obligations, these different types of obligations are enforced by validating and verifying different obligation states and state transitions.

2.4 Delegation of Authority

There are some papers [27][28][29][30], which try to extend XACML standard to support effective delegation of authority. Many of such existing literature are based on RBAC model. As in research [31], authors have proposed an Attribute-Based Delegation Model (ABDM) and its extension ABDMx. But in the core, it is also using role-based access control and lack of many features of DOA such as revocation. In Chadwick and Fatema's work [32], policy-based authorization is explained to secure critical data and protect the privacy of users. In this research, authors have utilized XACML Profile-based policies on data to achieve Human to Human delegation and administration. However, this monotonic delegation model lacks any provision for revocation which can bring lots of security challenges to the proposed model. These limitations make it incomplete and less fine-grained secure approach to facilitate delegation. In Tomaiuolo's paper [33], a generic open source framework for issuing and verifying delegation chains based on trust is

proposed. This revelation emphasizes the importance of a need for standardization and a common set of protocols to enforce delegation. Similarly, regarding workflow security, formal methods for delegation [34] in workflow management system is developed. However, it has not produced any tangible tool to support the claim regarding benefits of their approach. All these works are theoretical propositions and lack any proofs. Apparently, when these theoretical aspects are implemented in software, many real challenges emerge which are not considered.

CHAPTER THREE: SYSTEM REQUIREMENTS

Before presenting the object-oriented system architecture, we are required to specify the functional and security requirements which the model aimed to satisfy and then outline the core principles that will be implemented during the design process. The proposed system design is architected and designed in such a way that every functionality is highly configurable so that it can support modification in requirements in the future. This step is similar to the traditional software engineering practice, where the security features are often built in an ad-hoc manner. In this principle, design model (business logic) and security model are treated as different tasks. Based on this low-level system specifications, the overall system operational and security requirements are collected as shown in Appendix D.

3.1 Functional Requirements

Workflow system involves business process specifications that hold all the business logics as technical requirements. A business process involving some tasks needs to function effectively to meet its business goals. The primary purpose of any business processes is to increase customer satisfaction and reduce costs for an enterprise. The functional requirements help us to find out those core business values and describe overall operational processes of a business model. System functional requirement analysis is done based on UML models such as Use Case and its extensive textual descriptions. The object-oriented software development process begins with detailed UML diagrams where system

requirements are expressed in use cases. A use case is a graphical methodology used in system analysis to identify, clarify, and organize system requirements. The UML use case is used to explain a set of interactions between systems and users in a particular environment to achieve a specific goal. In GPWFMS environment, regular workflow system activities like to create, update, submit, delete, update, sign, delegate, revoke, approve, disapprove, withdraw, and archive a proposal during various phases are the typical proposal workflow functional needs. The proposal needs to be circulated to Department Chair, Business Manager, Dean, IRB, and University Research Administrator, University Research Director for review, approval, and signatures. Reasonable modifications to the proposal are permitted up to the submission. But, we need to make sure those tasks are visible only to the authorized users at any given time.

Using UML specification, we can identify all possible subjects, resources, and actions for our application. The behavior of use case is usually described in natural language, and these informal descriptions explain the allowed and denied accesses of actors to the system. For example, Department Chair is authorized to Approve, Disapprove, Delegate and Revoke actions and each action also includes Notify event. Overall functional requirements of GPWFMS can be generalized as shown in Table 1.

Table 1 Generalized Use cases for GPWFMS

Use case	Description
1. Create/add a proposal.	Allow the user to create/add a new proposal to the system as Principal Investigator (PI).
2. Delete the proposal.	Allow PI and Research Director to delete the proposal.
3. Update the proposal.	Allow PI and Co-PIs to update the proposal.
4. Submit the proposal.	Allow PI and University Research Administrator to submit the proposal.
5. Delegate the rights.	Allow Department Chair (delegator) to delegate his tasks (all or some) to Associate Chair (delegatee) from the same department.
6. Revoke the delegated rights.	Allow delegator to revoke the delegated tasks from the delegate.
7. Approve/disapprove the proposal.	Allow all authorized users i.e. Department Chair, Dean, Associate Chair (Delegated), etc. to approve/disapprove the proposal.

We have documented such functional requirements in use case diagram as shown in Figure 6, and their detailed textual descriptions are listed in Appendix A.

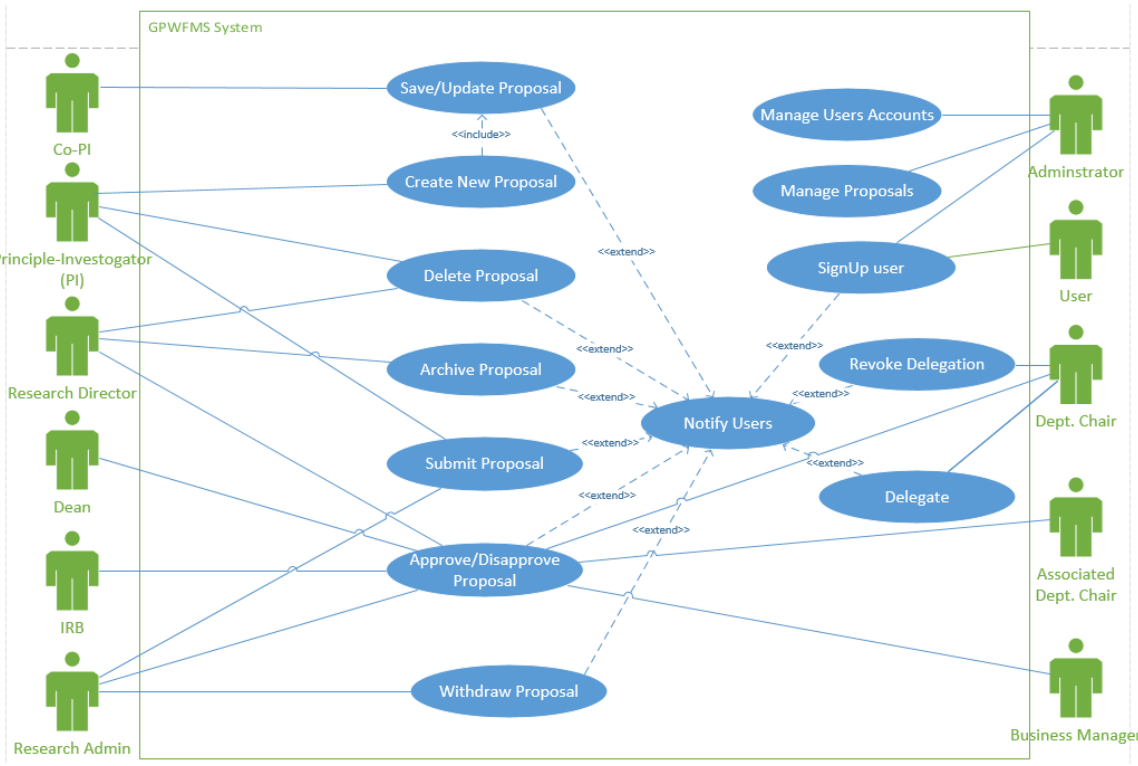


Figure 6 Use Cases for GPWFMS with Delegation of Authority

These use cases describe the mapping between actors and entities for an application to fulfill all its functions. A use case diagram corresponds to one low-level view of a model of a system where every object is regarded as protected and every access to an entity that is not part of use case is considered unauthorized. Such use cases implicitly define an access policy that adheres to the principle of least privilege [35].

To function properly, GPWFMS provides user management that helps to manage users, their corresponding position details, and other personal information. Similarly, it also requires having proposal management section where the user can search for their associated tasks and make changes to them from a central location. To fully function delegation features, it needs to provide a unified way to handle Human to Human delegation services called as delegation management. Also, a customized notification

service is desirable which will alert and also sends an email to the corresponding users about any changes to their proposal and requires their attentions. These desired specifications usually help us to understand the business processes and how can we automate them by driving interactions between various participants and the system.

For example, once PI submit the proposal to Department Chair for approval then the “Submit” button should not be visible to the same user navigating same proposal next time, whereas the “Approve” button should be displayed when Department Chair logs into the system. Developers can perform this kind of assertion in code level with writing lots of conditional statements. Such hard-coded security statements introduce complication making the application more rigid to any future changes and maintenance tasks in business logic. At the same time, it increases lines of code that introduces more application-dependent errors and security loopholes in an application.

Such low-level system requirements ensure a consistent model for development and allow a developer to break down the monolithic applications into smaller modular services that can interact with each other. In GPWFMS, such RESTful (Representational State Transfer) Application Programming Interfaces (API)-based services are designed in such a way that it supports the pre-defined functional requirements in an efficient manner ensuring a high-quality business application. However, in these functional specifications, the underlying security measures of the services are not considered.

3.2 Security Requirements

Security is the most powerful and efficient measure in software design to make it more robust and secure. It is equally necessary to implement and enforce non-functional security-related features in any application along with system function. However, one

problem with security requirements is that developers do not have expertise knowledge about secure software development. Beside this, security requirements are difficult to analyze and model [36] [37].

The potential attacks can occur not only from outsiders but also from within the system by the users who can misuse their assigned privileges. Such insider attacks can range from violating compliance goals, revealing confidential information, or altering workflow behavior. Therefore, to make the workflow system more secure along with fulfilling all the functional requirements, we need to assure that the proposed model meets essential goals of secure software design as explained below:

1. Confidentiality: The tasks (both normal and delegated) should not be disclosed to any other user. The private or sensitive personal and proposal information needs to be protected from unauthorized access or modification.
2. Integrity: Only authorized user can view whom the tasks that are assigned. Additionally, each activity needs to be validated and authorized as well as all the obligations (both pre and post responsibilities) accompany with that task must be enforced and fulfilled by the user during this process otherwise is not allowed.
3. Availability: All the tasks assigned to the user by the system or by another user (delegation) need to be visible and accessible to the user. Unless that privilege is forcibly revoked, expired or corresponding business rules are changed by the policy administrators.
4. Accountability: As proposal workflow involves many authorization actions that have access to sensitive data, proper caution needs to be taken to ensure that all

the measures are recorded and logged. These audit logs will help to back-track activities quickly and can be very helpful during the forensic investigation. Therefore, the system requires providing facilities to easily create records and reports describing sensitive information including who and when any particular data was accessed.

3.2.1 ABAC

The least-privilege policy that is implicitly defined by use case specifications may not be sufficient to counter all security risks. ABAC using XACML access control policy is strongly recommended [38] to achieve above-mentioned high-level functional and non-functional (security) goals. First of all, the processing model needs to be identified and enforced an access control policy at the service side. These policies are defined and written according to the business standards and provide the guidelines for access control to the system. The information modeled in the requirement analysis phase can be immediately used to generate fine-grained ABAC policies. These requirements can be translated into plain English format as listed in Appendix E that makes the business rules easy to understand and translate into access control policy.

A simple access control rule can be expressed in the human readable format as:

A "Tenured/Tenured-track faculty" is allowed to add a new "Whole Proposal".

We proposed a bottom-up approach for more refined security based on attributes held by each user and resource in an organization. With ABAC, we can easily add any additional context using various attributes (i.e. Subject, Action, Resource, and Environment or user defined attributes, etc.) to any request while a user is trying to access a resource. The final decision is based on information about the subject, resource,

environmental, and more hidden contextual information, that are often expressed as attributes and their corresponding values. An attribute is a property of an object; an authorization credential is a statement or assertion about an attribute. In particular, a credential must be based on defined attributes for a subject and during each action which validates and matches the pre-defined policy constraints. Restrictive authorization and administration can be handled by the implementation of XACML security policies based on these attributes; that can establish who can view, edit, and authorize specific parts of the proposal.

More detail metadata information about attributes and their corresponding category and potential values used in GPWFMS are listed in Appendix C. As shown in Table 2, for the defined access control rule, we can easily identify various attributes by looking into the pre-defined attribute metadata information.

Table 2 Attribute Dictionary Definition

Attribute	**Category**	**Type**	**Value**
position.type	urn:oasis:names:tc:xacml:1.0:subject-category:access-subject	http://www.w3.org/2001/XMLSchema#string	Tenured/Tenured-track Faculty
proposal.section	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	Whole Proposal
proposal.action	urn:oasis:names:tc:xacml:1.0:attribute-category:action	http://www.w3.org/2001/XMLSchema#string	Add

By comparing with the pre-defined attribute dictionary, for the given access control rule we can find out that the *subject* attribute is ‘position.type’ which has a value of ‘Tenured/Tenured-track faculty’, the *action* attribute is ‘Add’ and the *resource* attribute is ‘Whole Proposal’.

The above-mentioned system requirement can be tabulated as shown in Table 3.

Table 3 Requirement for Add proposal by Tenured/Tenured-track Faculty

Action: Add				
Rule	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligation**
Add Proposal by Tenured/Tenured-track Faculty	position.type = Tenured/Tenured-track faculty proposal.section = Whole Proposal proposal.action = Add			

The access control requirement as mentioned above specifies that the request of a Tenured/Tenured-track faculty to add a proposal will be allowed without any postconditions and attached obligation constraints. This kind of security requirement shows a normal access attempt by a user holding some pre-defined attributes to perform some actions on a secure resource.

XACML policy is written based on the pre-defined mapping between attribute metadata and XACML attributes in the access control rules. This pre-defined attribute information is used as a dictionary and is used to perform lookup during request creation and response validation. Such support for metadata of attributes makes the design flexible

to support federated attributes that are usually distributed in heterogeneous, distributed environment.

Such fine-grained access control produces a more secure and reliable system. Hence, it is desirable to have compact and adequate policies in the system that can satisfy all pre-defined requirements. These localized and unified security policies are used by the application to decide on any request from a user to perform actions on given resources depending on the provided contextual information. Using centralized security policies and mechanisms eliminates the tedious, repetitive, and labor-intensive manual procedures required to provision and manage security measures. The policy-driven system design will help to work seamlessly in its dynamically changing runtime environment.

3.2.2 Obligation

The proposed system needs to enforce any associated obligations before and after any tasks are performed. Severe security threats can occur if such constraints are not entirely implemented. There are two types of obligation *Pre-obligation* and *Post-obligation*; both are non-negotiable, and the system is required to enforce and apply them thoroughly. *Pre-obligation* which specifies the responsibilities a user/system need to fulfill before accessing and performing any tasks on a resource. On the other side, *Post-obligation* refers to the user/system's accountability after the action is either permitted or denied.

In GPWFMS, as we described in Table 4 and Table 5 below, access control rules can include one or both types of obligations. The pre-obligation constraint is to sign the proposal before approving it, and post-obligation is to notify via email with a rationale to the next person and all other associated users on the workflow. We can also classify obligations based on whether they will be carried out by the user or the system itself. For

example, the signing of a proposal is done by the user, so it is user obligation. On the other hand, sending an email is a system-level obligation performed by the system.

For example, another access control rule with having only post-obligation can be written in plain English format as below:

1. *"PI" can "Delete" a "Proposal" when SubmittedByPI = NOTSUBMITTED and not been already deleted without any pre-obligation but with Post-obligation: Send Email to all Investigators such as PI, CO-PIs, and Senior Personnel.*

Here, the *subject* attribute is 'proposal.role' which has a value of 'PI', the *action* attribute is 'Delete', the *resource* attributes are 'Whole Proposal' and 'READYFORAPPROVAL' and needed pre-conditions are that the proposal has neither already been submitted nor deleted. Additionally, the system needs to fulfill some obligation constraints to complete this authorization request successfully. As we can see, neither PI nor the system has any pre-obligation, but the system needs to enforce and satisfy the defined post-obligation requirement after successful access of the proposal. As identified in the given policy rule, the system needs to send an email to PI, Co-PI, and Senior Personnel (post-obligation) after deletion of the proposal.

The requirement as described above for deleting a proposal by PI is tabulated as shown in Table 4.

Table 4 Requirement for Delete proposal by PI

Action: Delete				
Rule	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligation**
Delete Proposal by PI	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED proposal.role = PI proposal.section = Whole Proposal proposal.action = Delete			System sends an email to PI, Co-PI, and Senior Personnel

The above-listed access control requirement describes the request of a user with proposal role of PI to delete a proposal that will be allowed if it is not yet submitted and removed. But while fulfilling this authorized access, the system needs to send an email to all associated PI, Co-PI, and Senior Personnel of that proposal.

Another access control constraint used in GPWFMS that involves both pre and post obligations as expressed and represented in the human readable format that follows:

2. *"Department Chair" can "Approve" a "Proposal" when ApprovedByDepartmentChair = READYFORAPPROVAL with Pre-obligation: Chair needs to Sign it first and Post-obligation: Send Email to all Investigators such as PI, CO-PIs, and Senior Personnel.*

Here, the *subject* attribute includes 'Department Chair', the *action* attribute is 'Approve', the *resource* attributes are 'Whole Proposal' and 'READYFORAPPROVAL' and needed conditions are that the proposal does not have any compliance information, and also all involved department chairs have already signed it.

The authorization constraint as mentioned above with obligations constraints can be listed as shown in Table 5.

Table 5 Requirements for Approve by Department Chair

Action: Approve				
Rule	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligation**
Approve Proposal by Department Chair	ApprovedByDepartmentChair = READYFORAPPROVAL position.title = Department Chair proposal.section = Whole Proposal proposal.action = Approve signedByAllChairs = true irbApprovalRequired = false	ApprovedBy DepartmentChair = APPROVED ApprovedBy BusinessManager = READYFOR APPROVAL	Department Chair signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Business Manager

The access control requirement as mentioned above stipulates that the request of a Department Chair to approve a proposal can only be granted when the proposal is waiting for approval. If all of the defined access control conditions are satisfied, then the system allows the user with position title of Department Chair to approve a proposal. Additionally, the system also needs to enforce and implement all obligations criteria. First of all, the user needs to sign the proposal, otherwise, approve action is not permitted (pre-obligation). After the user has signed the proposal, the system then must send an email to PI, Co-PI, Senior Personnel, and Business Manager (post-obligation). Also, the system needs to update the status of the proposal to indicate that it is now waiting for Business Manager Approval.

3.2.3 Delegation of Authority

Apart from standard functional features, GPWFMS also requires understanding the possibility of a potential obstruction in the workflow processes. GPWFMS allows some authorized users to delegate all or subset of their tasks/rights to another authorized person for completing the job on time. Besides, it needs to be flexible enough to support delegation requirements, such as delegation of authority, delegation of obligations, temporary delegation, transfer mode and revocation.

The key issue is evident in the real-world scenario such as how to model the DOA in which one user can hand over his/her authority to another user for a given period and allow for revoking that privilege afterward? In our proposed delegation model, we consider Human to Human delegation even though there are other forms of these delegations that exist including Human to Human, Human to Machine, Machine to Human and Machine to Machine [39]. The basic idea behind Human to Human delegation is that an authorized entity is allowed to forward his authority to another active object for timely completion of a task.

We tried to use and satisfy some of the salient characteristics that are mentioned in [39][40] to describe the behavior of our delegation model.

1. Monotonicity: “Monotonicity” defines the power possesses by delegator after delegation. For simplicity of design and implementation, we used Non-monotonic (Transfer) mode of delegation [41] in which delegator cannot use his delegated rights parallel with delegatee after delegation process. Since delegatee cannot delegate acquired permissions further, therefore our delegation model is limited to only one step delegation.

2. Permanence: “Permanence” describes the duration of delegated rights. In the proposed model, we allow the delegator to choose the limited date range so that delegation is only valid for that specified period. This temporary nature of delegation gives the system a level of security as the delegated policy will be inapplicable once the applied time is expired. Thus, manual revocation from the delegator is not necessary since auto-revocation is supported.
3. Totality: “Totality” characteristic defines the completeness of delegated rights i.e. partial delegation or total delegation [40]. In this model, we supported both types of totality features of delegation. If a delegator prefers the partial delegation, then the delegator can assign and select only a subset of access rights. This granular level permission based delegation refines the delegator's need. Such distinction of delegation rights allows the delegator to segregate the highly confidential tasks from others during delegation process and enable them to delegate based on trust level with delegatee. Such refinements prevent any unwanted risk of access due to handing over all available rights to delegator's subordinates.
4. Revocation: “Revocation” is used to take away delegated rights from delegatee in two ways namely; forced-revocation or auto-revocation [42]. Such revocation can be performed manually or automatically. In forced-revocation, a delegator can revoke delegated privileges any time whereas, in auto-revocation, delegated privileges automatically revoked upon expiry of duration. Both of such revocation is supported and implemented in the proposed delegation model.

Delegation is an essential and desirable feature in any modern enterprise. In the field of access control, it is extremely crucial to have a delegation that helps to simplify the

administrator tasks and to coordinate collaborative work securely, especially with the increase in shared information and distributed systems. Apparently, the delegation of tasks/rights to another authorized user is a very useful real-world situation by which workflow continues to successful completion even in unwanted situations like user or resource unavailability or overloaded with tasks. The delegation of authority is an essential business requirement in an enterprise or organization where different users need to perform dynamic business processes in a heterogeneous computing environment. Without DOA, tasks cannot be divided among users which result in the individual user being overloaded with pending tasks.

The delegation need is based on business rules and can change over time, which can be stored in static delegation policy. For such dynamic transfer of responsibilities, the system needs to allow adding new dynamic delegation policy in the policy repository at runtime. Hence, the system needs to be secure enough to support and reflect dynamically added delegation policy rules. The proposed delegation model needs to support both static and dynamic access control policies.

However, to model delegation constraint into a real-world software is a challenge, as it brings lots of complexities, risk and privacy issues associated with individual user's privileges and permissions. This decentralization of authorization can impose severe security risks to the organization by exposing high-level privileges to individual users. As delegation can cause a critical security threat to a workflow system, provision and mitigation approaches need to be implemented on any WFMS.

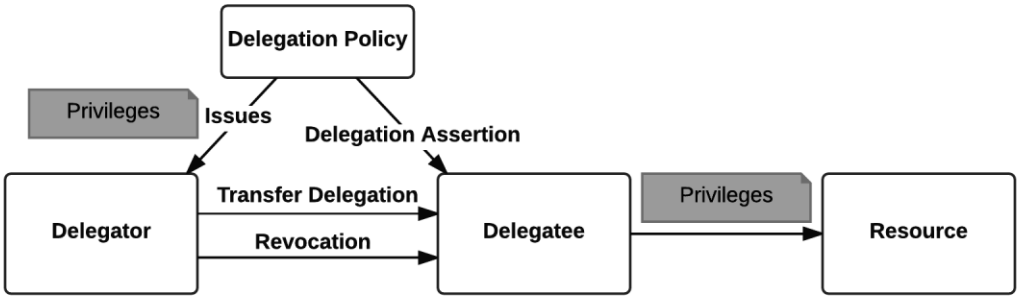


Figure 7 Conceptual Delegation Model

As depicted in Figure 7, this delegation model constitutes of interactions among delegator, delegatee, resources, access rights/privileges. The delegation of tasks among users is a powerful technique for managing the complexity of modular and adaptive applications. The primary requirement of any delegation model is to specify who can delegate what? For example, a delegator can delegate only a subset of his rights at a given context.

A delegation of authority is a suitable approach for handling such exception cases. The proposed model needs to tackle such break-the-glass scenarios as they can have security implications. This feature allows the authority to ensure alternative execution routing path to the workflow process that makes WFMS more flexible and efficient. An alternative route makes the workflow continuous and unobstructed even in the absence of a particular user at any stage. This feature helps the organization to fully utilize the available resources by allowing users to provision, manage, and de-provision their privileges. Trust gives a notion of achieving such security constraints [42]. If the given trust level is exploited, then that can be the point of security attacks and poses a threat to the whole system.

An example of a static delegation rule used in GPWFMS is expressed and represented in the human readable format as:

"Department Chair" can "Delegate" his actions "Approve/Disapprove" to "Associate Chair" from his own Department when ApprovedByDepartmentChair = READYFORAPPROVAL.

Here, the *subject* attribute includes 'Department Chair', the *action* attribute is 'Approve' and the *resource* attribute is 'READYFORAPPROVAL' without any further constraints.

The above-mentioned complicated delegation scenario from GPWFMS can be illustrated in use case as shown in Table 6.

Table 6 Use Case for Department Chair Delegates Associate Chair.

Use case #	UC-6
Use case name	Department Chair Delegates Associate Chair of his/her own Department.
Actors	Department Chair
Goal	Department Chair Delegates Associate Chair of his/her own Department.
Preconditions	1. The actor has an account on the system. 2. The actor's position title must be Department Chair.
Main Flow	1. The actor logged into the system. 2. The actor selects "Delegation" menu. 3. The actor selects "Add New Delegation" action. 4. The system receives the actor request and redirects the user to the new delegation page.

	5. The actor selects the "Delegate To" that is bind to delegate users based on policy rule specified such as User with position title of Associate Chair from his/her own Department. 6. The actor selects the "Delegate Actions" by selecting multiple checkboxes based on a policy defined for current actor's context. 7. The actor selects the range of temporal delegation period using starting date and ending date for the delegation. 8. The actor fills the reason for the current delegation. 9. The actor saves the delegation information. 10. The system sends notifications to the selected delegatee and current delegator. 11. The system records that on the delegation audit log.
Post-condition	1. The system saves the delegation with correct data submitted by the actor. 2. The actor can access the delegation for edit and revocation.
Alternative Flow	NONE
Exception Flow	NONE
Recovery Flow	NONE

The above-mentioned delegation scenario for the workflow system can be illustrated as shown in Figure 8. The *Department Chair* is allowed to delegate all or a subset of his access rights/tasks (such as Approve/Disapprove Proposal, etc.) to the *Associate Chair* that is defined in static delegation policy. In a general case, there are no

rules for Associate Chair in the system as there are no pre-defined access control rules for users in that position title. Apparently, when the *Associate Chair* tries to perform any of these tasks, he is denied access.

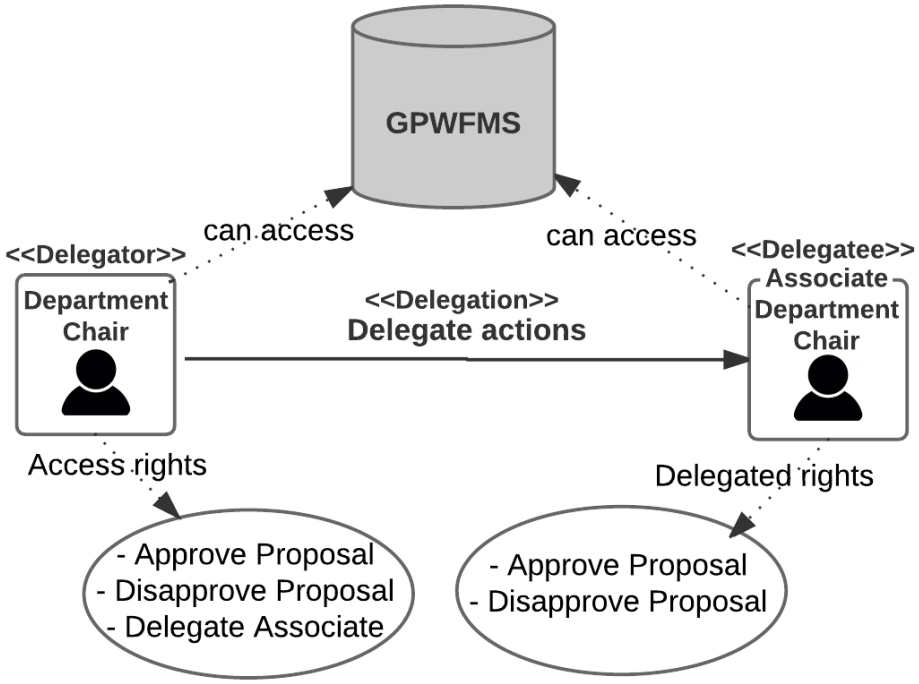


Figure 8 A simple example of Delegation Process in GPWFMS

For instance, let's consider the *Department Chair* from the Computer Science department wants to go on a vacation for a specified duration of time. The challenge is "What will happen to any proposal that is waiting for his approval?" Such unforeseen situations indeed lead to obstruction and unwanted delays to the overall flow of the system and can hinder the overall business goals. Therefore, to mitigate this exceptional situation, he is allowed to delegate a subset of his available tasks to the *Associate Chair* from his department. In such a scenario, he gives his subordinate his trust and permission to carry out the necessary actions. Also, he can revoke this temporarily delegated rights from his

assistant once he comes back or anytime he wants. Delegation and Revocation are important concepts that are essential for modeling and reasoning dynamic distributed systems. Such delegation of authority feature is desired in any adaptive and dynamic workflow system which provides proper document routing and real-time decisions making.

3.3 Access Control and Obligations in XACML

In our proposed system design, a policy administrator or generator, who understands the organization's security needs and business goals, can design customizable, XML-based access control policies, and host them in a central policy repository. To maintain proper authorization between different users and resources in GPWFMS, we have designed and implemented a series of XACML policy rules as shown in Appendix F.

Access control policy contains business rules defining overall functional and security specifications of the system. Besides, this policy also describes all actions applicable and available to a user based on given contextual information.

For instance, the security requirement as explained in Table 3 can be declared as a general XACML access control policy rule without any obligations constraints as shown in Figure 9.

```
<?xml version="1.0"?>
- <Rule Effect="Permit" RuleId="AddProposalByFaculty-Rule1">
    <Description>A "Tenured/tenure-track faculty" is allowed to "Add" a "Whole Proposal"</Description>
  - <Target>
    - <AnyOf>
      - <AllOf>
        - <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Tenured/tenure-track faculty</AttributeValue>
            <AttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" Category="urn:oasis:names:tc:xacml:1.0:subject-
              category:access-subject" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:position.type"/>
          </Match>
        - <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Whole Proposal</AttributeValue>
            <AttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" Category="urn:oasis:names:tc:xacml:3.0:attribute-
              category:resource" AttributeId="urn:oasis:names:tc:xacml:1.0:resource:proposal.section"/>
          </Match>
        - <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Add</AttributeValue>
            <AttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" Category="urn:oasis:names:tc:xacml:3.0:attribute-
              category:action" AttributeId="urn:oasis:names:tc:xacml:1.0:action:proposal.action"/>
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
  </Rule>
```

Figure 9 XACML Access Control Policy Rule without Obligations

Within such policy rules, we can define security constraints for each action so that each decision can reply along with required obligation needs. Then the application can quickly implement and enforce those obligation requirements. This new concept of constrained tasks to be followed before or after a request makes the software more secure and user more accountable.

For example, the access control specification listed in Table 5 can be converted into corresponding access control policy rule as shown in Figure 10.

```
<?xml version="1.0"?>
- <Rule RuleId="ApproveProposalByDepartmentChair-Rule13b" Effect="Permit">
    <Description>"Department Chair" can "Approve" a "Whole Proposal" when ApprovedByDepartmentChair = READYFORAPPROVAL and where condition check all
      department chairs are approved and no IRB is required.</Description>
  - <Target>
    - <AnyOf>
      - <AllOf>
        - <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Department Chair</AttributeValue>
            <AttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"
              Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:position.title"/>
          </Match>
        - <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Whole Proposal</AttributeValue>
            <AttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"
              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" AttributeId="urn:oasis:names:tc:xacml:1.0:resource:proposal.section"/>
          </Match>
        - <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">READYFORAPPROVAL</AttributeValue>
            <AttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"
              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
              AttributeId="urn:oasis:names:tc:xacml:1.0:resource:ApprovedByDepartmentChair"/>
          </Match>
        - <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Approve</AttributeValue>
            <AttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"
              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action" AttributeId="urn:oasis:names:tc:xacml:1.0:action:proposal.action"/>
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
  - <Condition>
    + <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
    </Condition>
  - <ObligationExpressions>
    + <ObligationExpression FulfillOn="Permit" ObligationId="sendAlert">
    + <ObligationExpression FulfillOn="Permit" ObligationId="sendEmail">
    </ObligationExpressions>
  </Rule>
```

Figure 10 XACML Access Control Policy Rule with Obligations

3.4 Delegation in XACML

Our proposed model classifies XACML server-side delegation policies into two main categories:

Static Delegation Policy: This includes global administrative rules that define who can delegate what kind of actions to whom. That sort of policy is based on the business logic of an organization and can change in future. Such rules define some special rights assigned to users that enable writing and directly influence effective delegation policy in the system at runtime. These delegation administrative constraints confirm that the delegated rights accessible to the delegator and transfer of such rights are allowed. For instance, the delegation requirement listed as use case description in Table 6, can also be expressed with static delegation policy rule as shown in Figure 11. This kind of administrative delegation rule allows delegators to create dynamic delegation rules about individual sets of resources.

```
<?xml version="1.0"?>
- <Rule RuleId="DelegateAssociateChairComputerScienceByDepartmentChair-Rule1" Effect="Permit">
    <Description>"Department Chair" of "Computer Science" can Delegate "Approve" and "Disapprove" actions to "Associate Chair" from the "Same
      Department"</Description>
  - <Target>
    - <AnyOf>
      - <AllOf>
        - <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Department Chair</AttributeValue>
            <AttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"
              Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:position.title"/>
          </Match>
        - <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Computer Science</AttributeValue>
            <AttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"
              Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:department"/>
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
  - <Condition>
    - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
        - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Computer Science</AttributeValue>
          </Apply>
          <AttributeSelector DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" Category="urn:oasis:names:tc:xacml:3.0:attribute-
            category:resource" Path="//ak:department/text()" ContextSelectorId="urn:oasis:names:tc:xacml:3.0:content-selector"/>
        </Apply>
      - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
        - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Associate Chair</AttributeValue>
          </Apply>
          <AttributeSelector DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" Category="urn:oasis:names:tc:xacml:3.0:attribute-
            category:resource" Path="//ak:positiontitle/text()" ContextSelectorId="urn:oasis:names:tc:xacml:3.0:content-selector"/>
        </Apply>
      </Apply>
    </Condition>
  + <AdviceExpressions>
</Rule>
```

Figure 11 Static Delegation Access Control Policy Rule

Dynamic Delegation Policy: This kind of administrative policy is created dynamically based on delegator's requirements and also includes a rule to support revocation of delegation tasks by the principle delegator. For instance, this shown dynamic delegation rule allows an Associate Chair (delegatee) to perform Approve and Disapprove tasks (Actions) on the proposal (Resource) from the same department (Computer Science) as an authority (delegator) on given delegation period. The delegator maps a dynamic relationship between a delegatee and a resource so that system understands the dynamic delegation rules.

```
<?xml version="1.0"?>
- <Rule RuleId="DelegatedApproveProposalRule3For-Associate-Chair-DelegatedBy-Computer-Science-Department-Chair" Effect="Permit">
    <Description>Edmund O. Schweitzer III of Computer Science with position title Associate Chair can "Approve" a "Whole Proposal" when Delegated by Computer
      Science Department Chair with position title "Department Chair" ApprovedByDepartmentChair = READYFORAPPROVAL and where condition check all
      department chairs are approved and IRB is required.</Description>
  - <Target>
    - <AnyOf>
      + <AllOf>
      </AnyOf>
    </Target>
  - <Condition>
    - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
        - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Approve</AttributeValue>
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Disapprove</AttributeValue>
          </Apply>
          <AttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"
            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action" AttributeId="urn:oasis:names:tc:xacml:1.0:action:proposal.action"/>
        </Apply>
      + <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-equal">
      + <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-equal">
      + <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal">
        - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only">
            <AttributeSelector DataType="http://www.w3.org/2001/XMLSchema#dateTime" MustBePresent="false"
              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" Path="//ak:currentdatetime/text()"/>
          </Apply>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#dateTime">2016-09-01T00:00:00-06:00</AttributeValue>
        </Apply>
      - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal">
        - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only">
            <AttributeSelector DataType="http://www.w3.org/2001/XMLSchema#dateTime" MustBePresent="false"
              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" Path="//ak:currentdatetime/text()"/>
          </Apply>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#dateTime">2016-10-01T00:00:00-06:00</AttributeValue>
        </Apply>
      </Apply>
    </Condition>
  - <ObligationExpressions>
    + <ObligationExpression FulfillOn="Permit" ObligationId="sendAlert">
    + <ObligationExpression FulfillOn="Permit" ObligationId="sendEmail">
    </ObligationExpressions>
</Rule>
```

Figure 12 Dynamic Delegation Access Control Policy Rule

CHAPTER FOUR: SYSTEM DESIGN AND IMPLEMENTATION

The basic concept for workflow-enabled applications is the association of executable tasks with each step in the business process. To develop guidelines for the design of a workflow, we first need to understand an overview of the organizational needs that need to be satisfied in the workflow life cycle. Our proposed system design provides a holistic approach for implementing attribute-based access control in Service Oriented Architecture (SOA). With the rising popularity of distributed systems, the management of workflow for an organization, which involves different levels of users and resources, needs more time and effort. High level of collaboration and information sharing requires such systems to be more secure and reliable while utilizing all available organizational resources efficiently. Implementation and enforcement of secure access control mechanisms in an SOA environment are considered a complex challenge [43].

4.1 Architecture

GPWFMS is built based on SOA environment in which decoupled services interact with each other by exchanging a standardized REST-based message format without consideration of the underlying implementation. It involves presentation, business logic, data access, and data storage layers. The user is provided with a generalized and user-friendly interface that acts as the top-most layer of the application, which translates the response from the system to a readable format. The logical business layer processes and communicates data between the layers. This middle layer provides building blocks for

aggregating loosely coupled or decoupled services as a sequencing process aligned with business goals. The data storage layer consists of a NoSQL database that allows persistent data access.

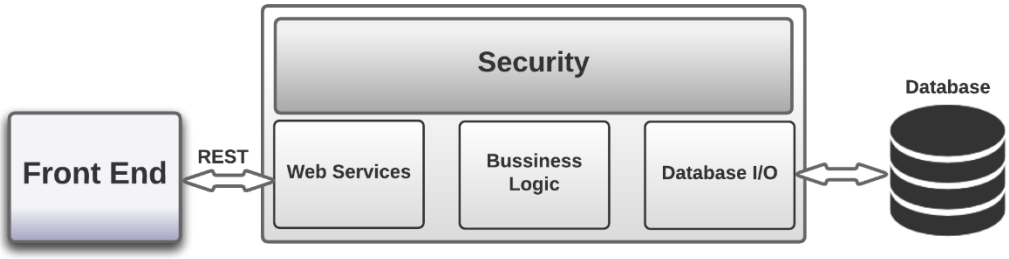


Figure 13 Modular Design of GPWFMS

As shown in Figure 13, GPWFMS is designed based on a scoped and modular approach for a typical 3-tier application architecture and to support the pre-defined system requirements. In this architectural pattern, the front end client can communicate with web services via REST call from the user interface layer. The business layer controls all functionalities of the application, and the data access layer allows the backend database to be connected with the application via a database Input/output (I/O) interface. We enforce policy based access control mechanisms in all three layers. Along with these steps, the system requirements for both security and functional are implemented and validated.

A representative block diagram of the authorization architecture employed in GPWFMS is shown in Figure 14.

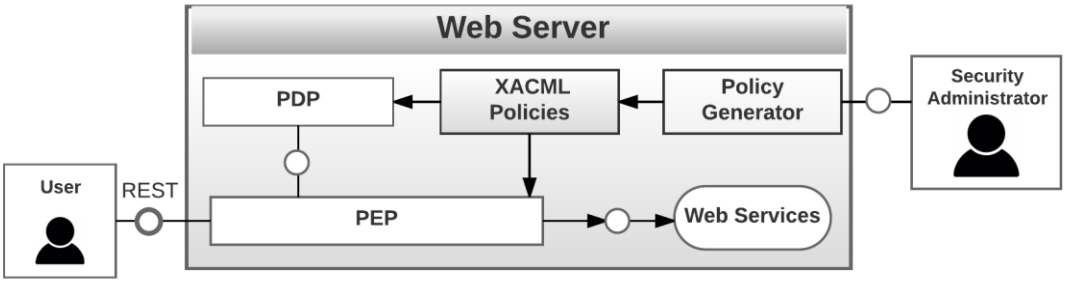


Figure 14 GPWFMS Block diagram

During development of any workflow systems, we consider the coordination of activities, resources, data, and applications. The component diagram shown in Figure 15 demonstrates the underlying interactions between various elements in our monolithic application. As shown in Figure 15, proposal management system requires a series of functions from the creation of a research proposal to the final submission. Specifically, the standard enterprise workflow functionality along with instant notification features with customizable and configurable user-friendly interfaces are designed. These services include various time-consuming and user-centric activities. Based on the workflow status of a proposal, it initiates an automated process and routes the document toward the appropriate users. This automation allows each user to quickly identify and view their current tasks along with the anticipated workload.

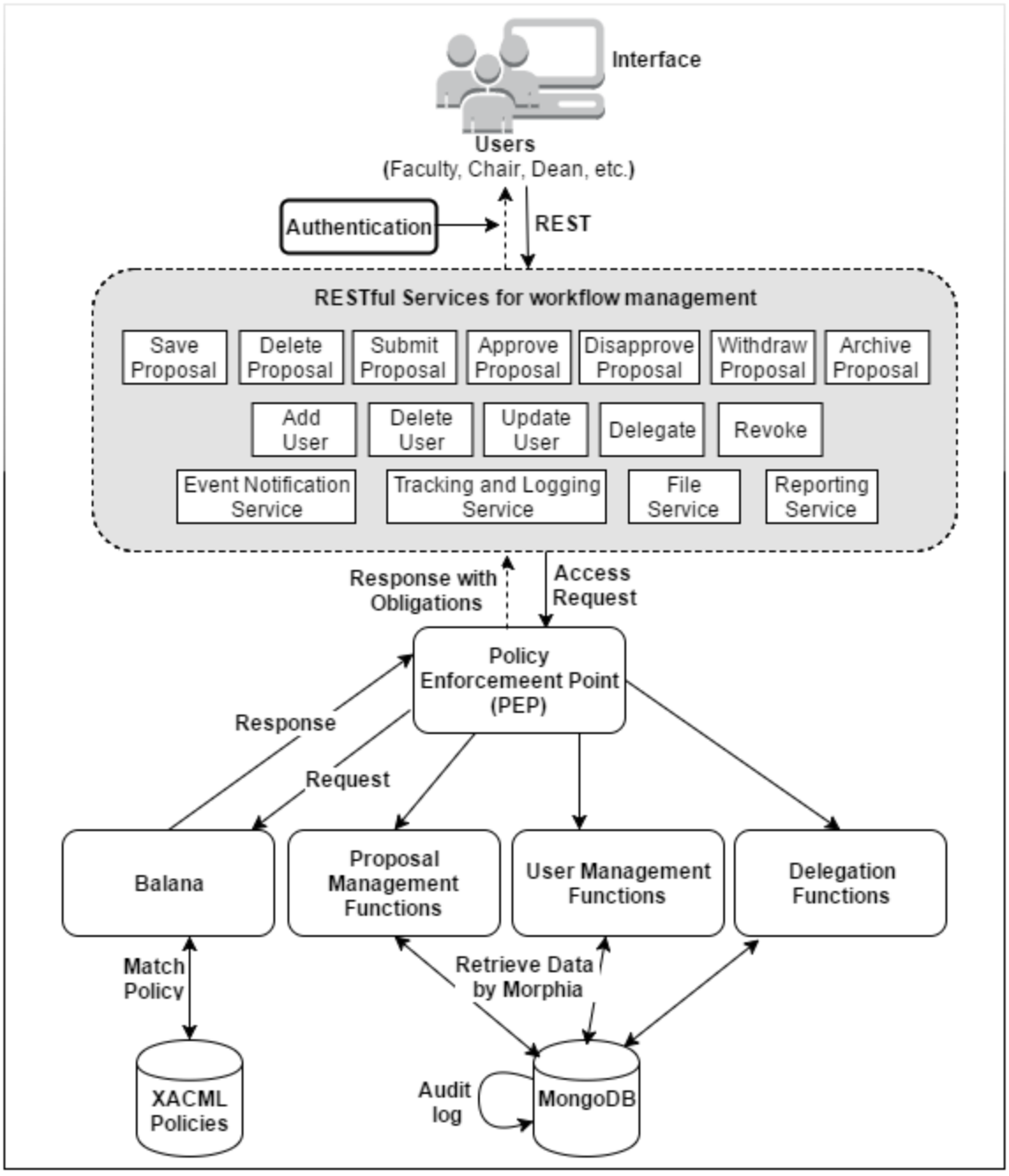


Figure 15 Application Architecture of GPWFMS

The user handling is carried out by 'User Management' functions that include services like adding, deleting, updating any user and their details. The proposal information is handled by 'Proposal Management' services that include many activities such as saving, updating, deleting, submitting, approving, disapproving, withdrawing, and archiving

proposal documents. The ‘Delegation’ features include services, like delegating and revoking delegation are also supported. The system needs to handle task automation automatically, for this functionality the ‘Event Notification’ service sends email alerts to users with notification of the changes to the proposal. The ‘Process Monitoring and Reporting’ functionality allows monitoring currently available documents in the system. This service also enables users to create reports containing detailed information on current workload, future workload, obstructions, etc. based on “historic” processing data. The ‘File Service’ allows the user to upload and download files from the system. During each step, information about ‘Tracking and Logging of Activities’ are recorded and logged onto the system that supports non-repudiation security requirements.

To achieve a goal of designing a loosely coupled workflow management system, GPWFMS uses the following tools and techniques:

4.1.1 RESTful Services

Software applications (especially popular web applications) are using open well-designed web services i.e. APIs, and using such public authorization services provides more interoperability among numerous distributed systems. API-driven REST based architecture allows having shared, on-demand and scalable services. REST is a stateless architecture which involves resources that are represented as Unified Resource Locators (URLs). The standard approach is to expose a set of web services to the rest of the world via the API Gateway. Such exposed web API endpoints permit any external applications to call the services of a workflow engine from outside the organizational boundaries. RESTful web services enforce a centralized and shared business logic across distributed system. REST can consume data streams in multiple formats such as plain text, XML and

JavaScript Object Notation (JSON). This flexible feature makes REST the ultimate choice for client-side development.

In GPWFMS, APIs are used to connect enforcement points which control access to sensitive information. Access control checks along with RESTful services help to prevent, detect and stop unwanted access to the system. Such web services are easy to develop and deploy. Additionally, they are usually lightweight, inexpensive to host and maintain. GPWFMS implements JAVA based RESTful web services (JAX-RS) to interact with the front-end client and backend database records via AJAX (Asynchronous JavaScript and XML). We create RESTful web services using the reference implementation of JAX-RS 2.0 i.e. Jersey which provides Client APIs to the front end.

Such RESTful web services abstract all the complex working mechanism of such access control by providing developers easy to use interfaces. This high-level abstraction allows developers to focus on business logic rather than understanding underlying complex security policies. The functional and security requirements are defined by XACML access control policies and using the XACML framework; the policy enforcement is implemented and achieved under this standard architecture. Additionally, each service is bound with underlying access control capabilities to make them more secure and to fulfill all functional and security requirements of the system.

Within the API level, the security authorization, authentication, and attestation is performed based on requested information and available XACML policies. However, one of the critical issues while using such publicly visible services is security. Any unwanted hackers can obtain user’s confidential information and can perform unauthenticated works via those public services. To prevent such unwanted risks, we need to increase their

security, reliability and to enforce access control based on user's requests. In GPWFMS, user authentication relies on the identity of the user from the login context where a unique session token is assigned to a particular user to validate their credentials and persist till they switch their account or logs off the system.

4.1.2 Database

Contained within this system is a database that stores relevant entity's information. To manage the attributes of every subject and object, they must have corresponding entries in a database that allows attribute retrieval and comparisons. The proposed robust architectural solution allows the system to generate, store and analyze enormous amounts of information with increased speed and scale. To overcome such data-driven requirements we choose, MongoDB[2] was the best suited No-SQL backend database.

Traditional 'relational' database models store information in hierarchical rows and columns in a tabular format. However, such mappings and relationships are impossible in complex datasets harvested from vast and concurrent data streams. MongoDB is more document-oriented because each document is stored as JSON objects and as attribute-value pairs. With a document like structure, it allows quick retrieval and faster processing of data while making it more readable and scalable for the user.

Four primary database collections are used in GPWFMS, namely *Users, Proposals, Notifications,* and *Delegations*.

The User database collection holds the detailed information of a user as well as login information necessary to authenticate the user into the system during login.

User information includes the following data:

[2] https://www.mongodb.org/

- User account data: A user account name and password.
- User detail information: A user's given names, contact information (such as addresses, phone numbers, and email addresses), and departmental position/role information.

The Proposal database collection contains sensitive information for a proposal, including various critical information related to it. A general proposal includes:

- Project information: Proposal specific information, such as the project type, title, and date related information.
- Financial information: Budget details, sponsorship information, and cost sharing information.
- Investigator information: Details about PI's, Co-PI's and senior personnel.
- Signature information: Signatures and notes from corresponding authorized users.

The Notification database collection stores information regarding recent changes to the data (user, proposal, etc.) and notifies the appropriate users.

The Delegation database collection contains information about the delegator, delegatee, delegated actions, duration of delegation and the reason for the delegation.

Additionally, to support Revocation of an individual delegation, each time a delegator assigns a delegation, a new dynamic delegation policy id is generated. The id is then added into the delegation *PolicySet* template at runtime. It is crucial to store dynamically created policy's id in the *PolicyId* attribute of dynamic delegation *Policy* node. The dynamic mapping between delegator and policy is also stored in the Delegation

collection so that revocation can be enforced based on the user authentication and the stored policy information.

4.1.3 Morphia

Morphia[3] is a lightweight library for mapping Java objects to and from the MongoDB database. Morphia is an Open Source Fluent Query API that uses annotations and standards to interact with code and database. It adds a layer of abstraction between Datastore and Data Access Object (DAO) of Java application that makes working with Java exceedingly comfortable with MongoDB. It makes working with data in Java easy as it creates a data persistence interface in between. Morphia is MongoDB's Java Persistence API (JPA[4]) which handles data access operations with less code. We can easily customize persistence and common data access patterns like Morphia's datastore and DAO as per application's need.

4.1.4 Balana

Balana[5] is an open source XACML Implementation by WSO2[6] that supports XACML version 3.0 specifications and creates Policy Decision Point instances that can be embedded in web service level.

4.2 Design and Implementation of Obligation Mechanism

The architected solution prospect of the model is comprehensive and extensive with the use of latest XACML specification. In XACML v3.0 specification, the underlying evaluation context model and the authorization decision request format is generalized.

[3] https://github.com/mongodb/morphia
[4] https://en.wikipedia.org/wiki/Java_Persistence_API
[5] https://github.com/wso2/balana
[6] http://wso2.com/

The total numbers of security access control rules defined in GPWFMS is shown in Table 7:

Table 7 Security Rules formulated for GPWFMS

No. of Access Control Rules	**No. of Rules with Obligations**	**No. of Static Delegation rules**	**Total**
93	49	4	97

XACML 3.0 Core specification supports obligations but does not distinguish between the different obligations types. Therefore, there is a significant need to extend the feature of XACML to support such obligations types. The latest obligation specification that is extended in XACML 3.0 defines that each definition of the obligation contains a unique identifier and can include zero or many lists of parameters, each with a locally unique name and data type. XACML allows us to describe an obligation method and its parameters as an attribute assignment so the actual definition of its syntax and semantics can be implemented quickly. Even though the XACML policy language is very flexible, there is currently no generic method to specify the obligations send from PDP to PEP. There is no standard conceptual model for obligations and their enforcement. Obviously, conflicts may arise among a set of responsibilities that require the need to keep account of relations between obligations for accuracy. The PEP is responsible for decoding and checking each response for any obligations constraints and negotiates to enforce the embedded constraints. Finally, PEP keeps track of the obligations' state and imposes the restrictions. Although this is an important issue, especially to support privacy, advanced tracking of data flow is quite neglected and not properly handled by XACML.

Depending on the nature of the obligation, it can be viewed as an additional restriction on the access right. An XACML obligation is an action to be performed before and after a particular event is triggered. Specifying obligations in access control policies is more secure and flexible than hard-wired in code-level. The ability to configure the obligation requirements externally in XACML policies enables a security administrator to activate or deactivate such security requirements dynamically without restarting or redeploying the running service. All associated obligations are replied along with the authorization decision in response to each system actions as shown in Appendix H. The actual interpretation of these obligation constraints is done by the developers and can be easily enforced and implemented in the client code.

```
<?xml version="1.0"?>
- <ObligationExpressions>
  - <ObligationExpression FulfillOn="Permit" ObligationId="sendAlert">
    - <AttributeAssignmentExpression AttributeId="obligationType">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">preobligation</AttributeValue>
      </AttributeAssignmentExpression>
    + <AttributeAssignmentExpression AttributeId="signedByCurrentUser">
    + <AttributeAssignmentExpression AttributeId="alertMessage">
    </ObligationExpression>
  - <ObligationExpression FulfillOn="Permit" ObligationId="sendEmail">
    - <AttributeAssignmentExpression AttributeId="obligationType">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">postobligation</AttributeValue>
      </AttributeAssignmentExpression>
    + <AttributeAssignmentExpression AttributeId="emailBody">
    + <AttributeAssignmentExpression AttributeId="emailSubject">
    + <AttributeAssignmentExpression AttributeId="authorName">
    + <AttributeAssignmentExpression AttributeId="piEmail">
    + <AttributeAssignmentExpression AttributeId="copisEmail">
    + <AttributeAssignmentExpression AttributeId="seniorsEmail">
    + <AttributeAssignmentExpression AttributeId="chairsEmail">
    </ObligationExpression>
</ObligationExpressions>
```

Figure 16 Obligations Expression Format

A rule or policy or policy set may contain one or more obligations. In GPWFMS, we have 49 access control rules that include obligations (either per/post or both) attached to them as shown in Appendix D and E. As seen in Figure 16, the scope of an obligation expression in an XACML rule is bound to the target and condition of the rule containing it. Such obligation requirements can be associated with both Permit/Deny decisions as specified in the *FulfillOn* attribute of obligation expression. During the evaluation, when

the effect of the policy or policy set matches the value of the *FulfillOn* attribute of the obligation, then only that requirement is returned along with the authorization result. To support two different types of obligations, we define XACML policy rules with *AttributeId* attribute with value *obligationType* for the first obligation expression element as shown in Figure 16. To denote pre-obligation, we assign the attribute with the string value of *preobligation* whereas to denote post-obligation we assign the value of *postobligation*.

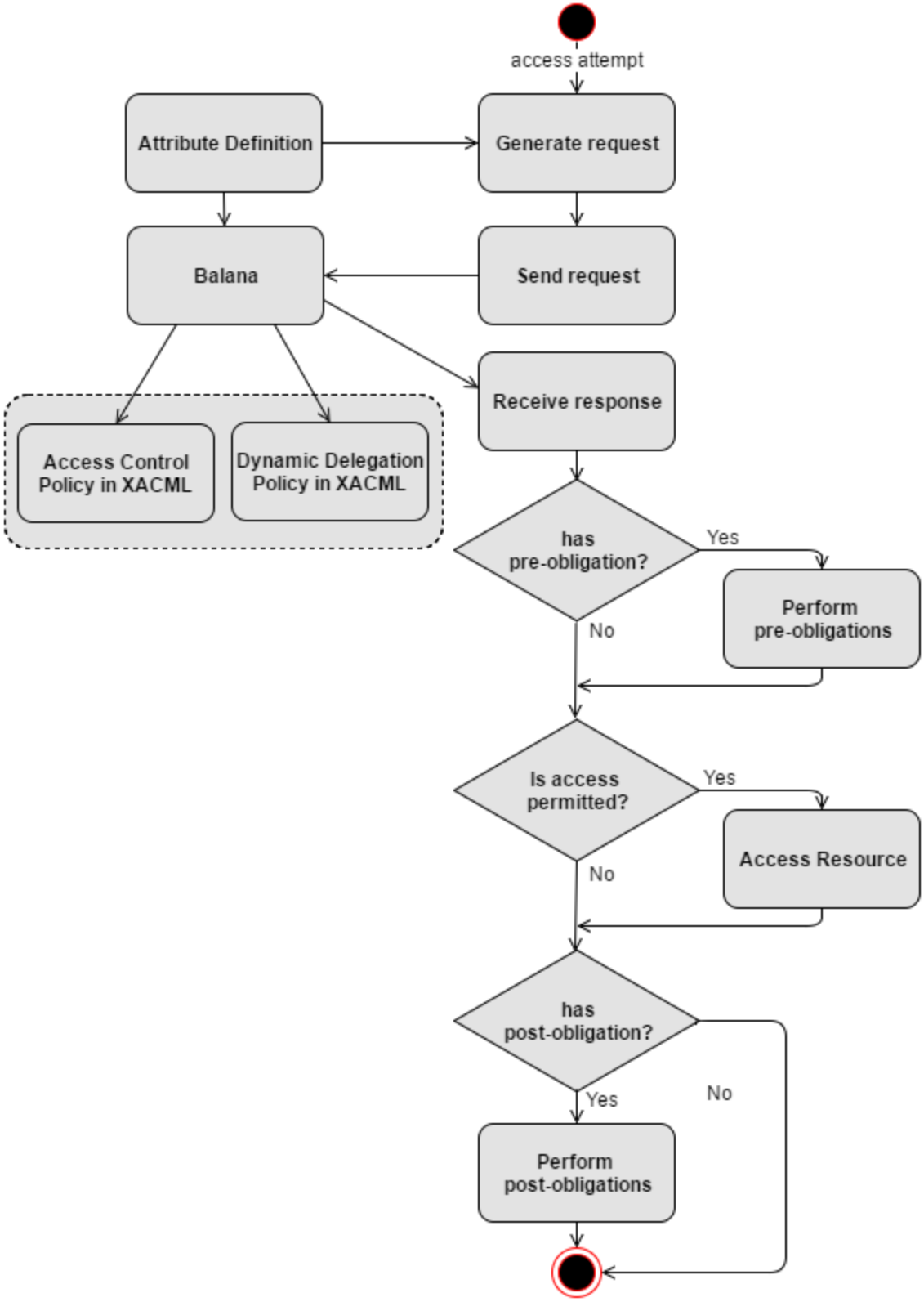


Figure 17 Obligation processing in GPWFMS

The overall control flow for any regular access attempt in GPWFMS is illustrated in Figure 17. For every access attempt, our model generates the request based on metadata information of attributes. It then sends the generated request, as shown in Appendix G, towards Balana to validate the attributes value and to determine the authorization. Balana looks for any match in Access Control and Dynamic Delegation Policy. Whenever the matched policy is found, those attribute values are also returned from Balana to the application as a combined decision with results as shown in Appendix H. If the authorization decision includes pre-obligations, those requirements are enforced and performed by the application first. If all pre-obligations are correctly executed, and the decision is Permit, then the system allows the requested access to the secure resource. If this response results in Deny decision, then the system prohibits access to the resource and tries to check if it includes any post-obligations in authorization decision. If no such obligation constraints exist, then the system follows the normal workflow path. This control flow is also applicable to delegation based access request from a delegatee to access a resource. In such a case, Delegation of Obligations needs to be fulfilled and enforced by the application as each delegated task can also bear some obligations to the delegatee and the system.

4.3 Design and Implementation of Delegation Mechanism

Our workflow system will provide any delegator with a user-friendly web interface as shown in Figure 18. The given screenshot shows the delegator can specify all delegable users and tasks to be delegated via provided unified user-friendly interface. This policy-driven delegation provides an abstract view that hides the details from the delegator about the complexity of delegation access control policies. This centralized interface allows the

delegator to see, grant and revoke access rights in an easy and unified way without understanding the underlying technical details. However, the actual mapping from the high-level abstract view to the low-level access control is handled by the proposed delegation model.

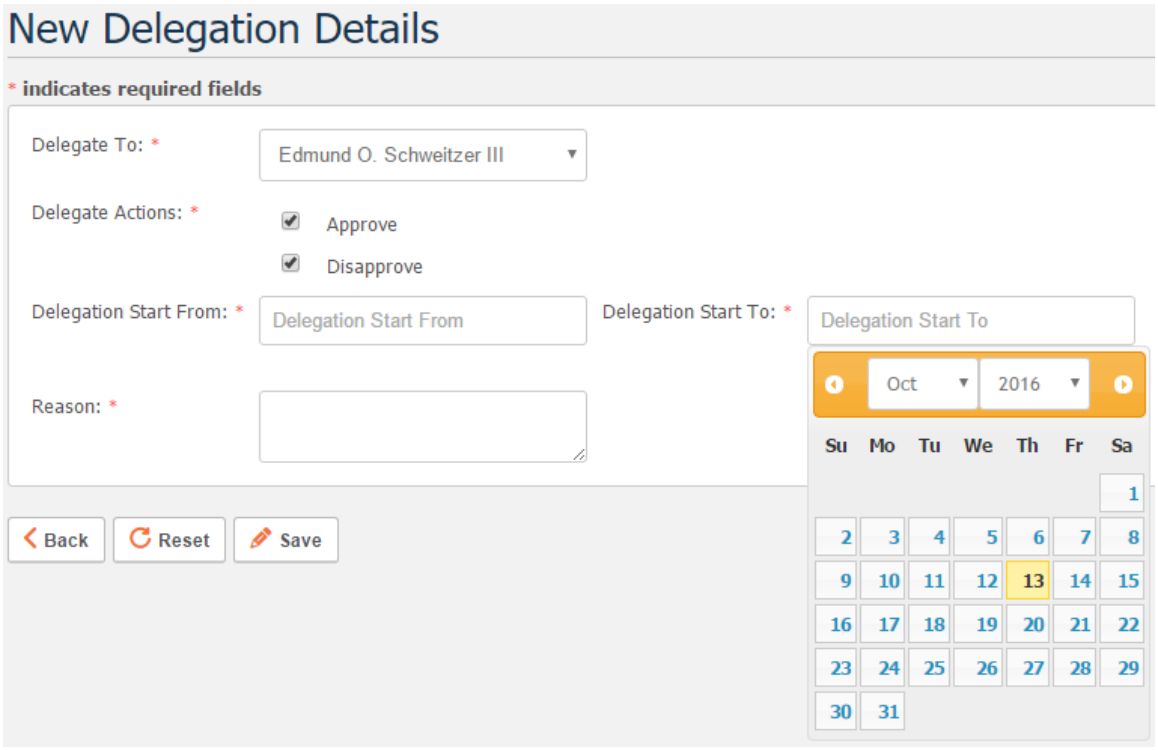


Figure 18 Delegation User Interface

In RBAC, it demands a significant number of delegation be created and managed with the number of roles and resources increase. However, this can be minimized by using the ABAC model which reduces the complexity of security administration. Policies based on security constraints fully control the proposed delegation model, thereby reducing the code level conditional ‘if-then-else' implementation. Assignment of the delegation are based on time, workload and users’ attributes. Often such delegations are short-lived and come into play when certain conditions are satisfied [34]. Based on delegator’s

requirements, effective administrative policies are generated, as shown in Figure 12, and dynamically added to the policy repository.

Using delegation the global administrators/authorized users can provision some constrained administrative/user rights to the local administrators/users. The dynamic and decentralized delegation distributes the privileges that make the workflow more flexible and scalable. The given system supports dynamic delegation that can create delegation policies on the fly without the need of redeployment of the application. Authority is often granted to an alternative subject if the primary subject is absent for an extended amount of time. Such situation can interrupt the normal businesses workflow hence someone must be available to act on the former's behalf. This scenario typically occurs when there are not enough users to process the workload or if a user wants to offload tasks to their subordinates. For such situations, it is necessary to add additional resources to the workflow system. Thus, by dynamic delegation, the workflow system offers the user the ability to change the routing process during execution, preventing obstruction of the workflow. While delegation is an important feature to keep pace with the dynamic nature of business, it is necessary to monitor and assure that none of the security constraints are violated. This model provides the delegation log facilities that can be very helpful for forensic investigations. During the provision of DOA, it should have minimal errors and ensures uniformity between all user permissions while making delegation a straightforward and risk-free activity.

In our delegation model, the delegation rights are differentiated from the normal access control rights. However, during evaluation, both access control and administrative delegation policies work together to generate a single decision. Underlying complex

processing of delegations is performed by our proposed model based on the control flow diagram shown in Figure 19.

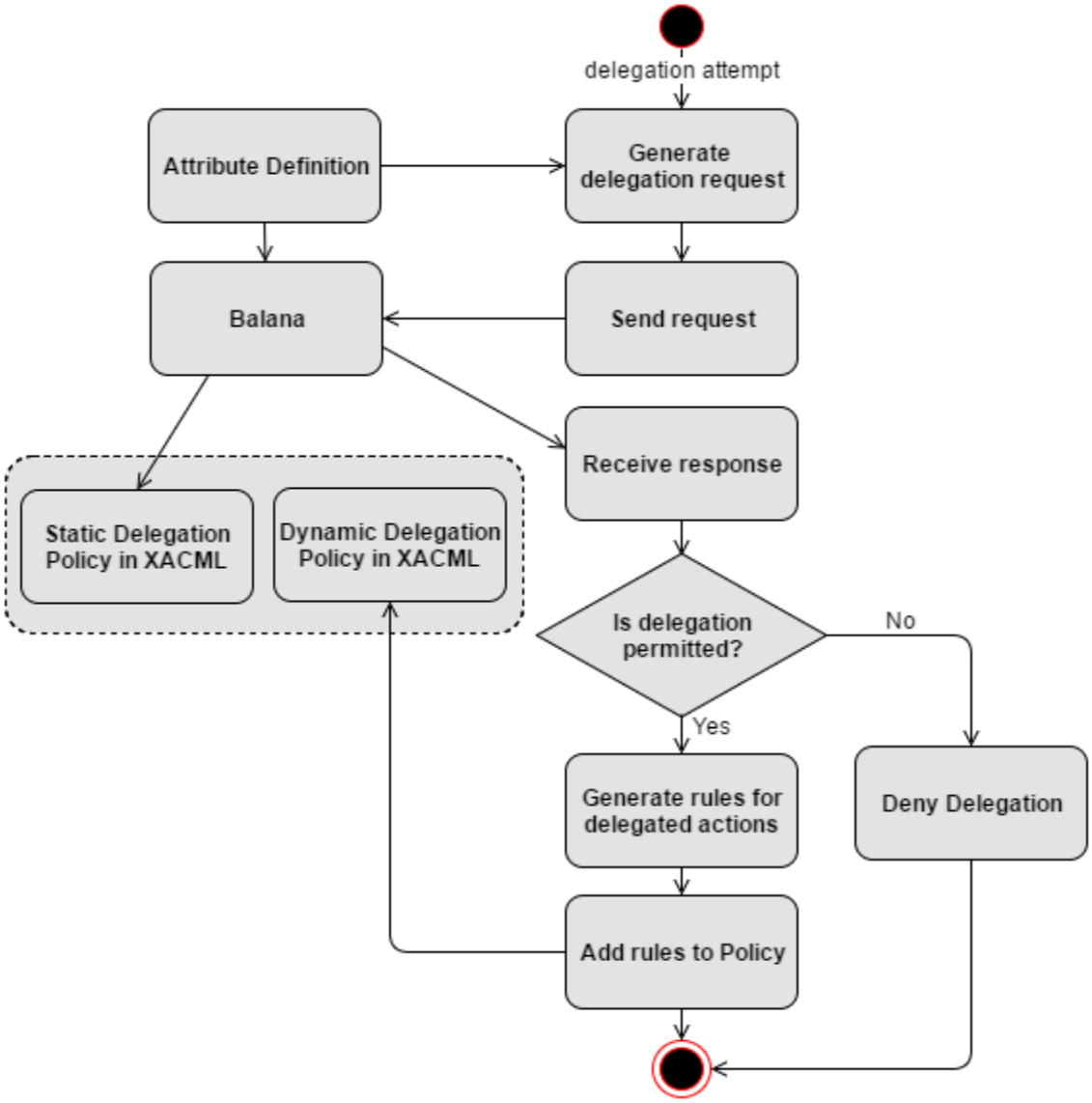


Figure 19 Delegation processing in GPWFMS

For each delegation attempt, the application generates a delegation based request that is validated using a pre-defined attribute dictionary. Once verified, the request is forwarded to Balana to process the request and attempt to match with an existing static delegation policy written in XACML format. During this process, Balana looks up

attributes and their corresponding values in attributes' metadata information. According to the evaluation, the resulting decision is replied, and the application receives the corresponding response. If the authorization decision is permitted, then the delegator's requirements dynamic delegation policy rules are generated for the delegated users and actions. Thus, dynamically created rules are added to the dynamic delegation policy set template so that a new delegation route for the delegatee. If the decision is to deny the delegation attempt, then the request is not fulfilled by the system.

Recent work [44] tries to add delegation extension to XACML 3.0 to express the right to administrate XACML policies within XACML itself using Administration and Delegation Profile. The delegation profile draft explains how to negotiate for the right to issue a policy, but has not provided any rules for removing a policy. In our proposed model, delegation is achieved by creating new dynamic delegation rules during the delegation process to define all access and delegation privileges in an XML format using XACML policy specifications. This effective delegation policy is automatically added to the policy store so that the system can directly reflect the changes at runtime.

We adopted a secure and flexible revocation model in WFMS, which gives a delegating user (delegator) power to revert the privileges from the one he has delegated (delegatee). Both delegation and revocation take account of time constraints, so our system must account for this provision. As delegation can cause a critical security threat to a workflow system, provision and mitigation approaches are implemented on GPWFMS using XML based policies. The solution provides a rule for both forced or auto-revocation methods to the delegator to avoid any uncontrolled delegation propagation to the delegatee. In our model, revocation can be performed automatically when the delegation context is

no longer active, or manually by an authorized user i.e. delegator. User revocation is performed by allowing and deleting dynamically generated delegation policies from the policy repository.

CHAPTER FIVE: VALIDATION AND EVALUATION

The new proposed design model implemented in GPWFMS need to be properly tested and evaluated based on all pre-defined system specifications. The model needs to be tested using test cases that reflect a particular business function. To do so, we need to select some specific test criteria which define possible inputs data and test oracles for verification.

5.1 Testing

Testing is a crucial step to analyze and evaluate the design implementation by developers. It intends to assure the quality of an application by finding defects or any security vulnerabilities that may have been introduced at the code level. Often developers are required to build their test harnesses based on business scenarios. The system should incorporate mechanisms to verify the API behavior using a set of appropriate testing tools and techniques. Therefore, to build the GPWFMS according to our requirements and free of errors, proper continuous testing is carried out. The pre-defined system requirements act as the acceptance criteria for GPWFMS.

The following definitions are used to clarify the distinction between functional and security policy testing.

1. Functional Testing: This involves generating and executing test cases based on the use cases and business requirements. For example, a faculty can add a proposal.

2. Security Policy Testing: It involves the activity of designing and evaluating test cases governed by written access control policy. The primary focus of security policy based testing is to explore many security flaws as possible. For example, the faculty must be either Tenure or Non-tenured track to add a proposal.

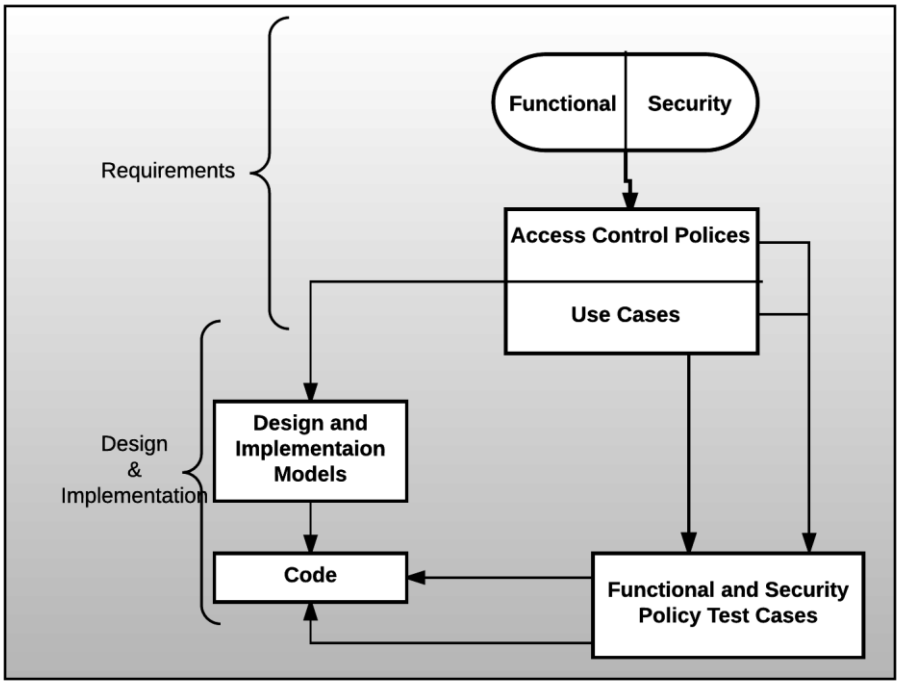


Figure 20 Testing Model in GPWFMS

As illustrated in Figure 20, the overall testing steps are performed to verify the design and implementation compliance with the pre-defined system specifications, e.g. functional and security requirements. System requirement testing involves testing of both operational and security policy that encompasses security as well as functional requirements. The pre-defined specification documents defined in UML diagram includes the technical description of the mainstream workflow scenarios as well as other non-functional security concerns.

System requirement testing is an essential step as it involves the testing of implementation of the proposed design. By thoroughly examining the system, it provides enough confidence in the system design, functional, and security implementation. Also, repeated and adequate testing ensures that our developed application contains high-quality codes.

Examining all the logical rules with proper implementation using manual inspection is a lengthy and tedious task. Therefore, it is always desirable to automate the process with the help of test cases and scripts. The automatic generation and execution of test cases are obtained using Selenium WebDriver[7]. Our testing methodology uses a combination of Selenium IDE[8], Selenium WebDriver, and JUnit[9]. Selenium IDE is a firefox browser plugin that records user actions on the visible aspects of an application. On the other hand, Selenium WebDriver is an Object-Oriented API that supports Data Driven Testing and Cross Browser Testing for test cases created using element locators and WebDriver methods/commands. In contrast to time-consuming and tedious manual testing, test automation tools such as Selenium allow verification of all possible workflow and alternative scenarios in a repeatable manner. The use of programming logic in each test case along with the overall flow of information allows complete testing of a secure workflow application. For a selection of test criteria, we select a particular test scenario during the workflow process, that involves both functional and security access control. Using Selenium WebDriver, a total of 53 different test cases are written, tested, and deployed that covers most of the mainstream workflow scenarios.

---

[7] http://docs.seleniumhq.org/projects/webdriver/
[8] http://docs.seleniumhq.org/projects/ide/
[9] http://junit.org/junit4/

These scenarios simulate various user tasks and business activities. Based on the test scenario, we validate the test oracle that defines the expected permission or prohibition for a particular action by the user. The testing and verifying the compliance of all access scenarios independently which makes the overall system design more secure and robust providing high levels of security. We use an incremental strategy to test scenarios such as the test cases, since such test cases are dependent on each other and can be reused. The security policy test cases are built in the complement of existing system level functional test cases. Hence, we can test and verify both requirements at the same time.

5.2 Results

Table 8 GPWFMS Test Results

	#Total Test Cases	**Test Result**
GPWFMS	53	Pass

The given Table 8, shows the overall test results from the automated testing. The results indicated that our all test cases have successfully passed. This result proves that our automated testing's coverage is high, almost all pre-defined system requirements that are mapped as access control policies are tested successfully and implemented in a secure manner. This result gives great assurance that our system's implementation code is operating correctly with good software quality and as desired on any valid input test data.

5.3 Threats to Validity

Complex business logics in XACML policy can be expressed in different ways. This high expressiveness results in a high degree of complexity and makes the evaluation of the policy in the enforcement step more difficult. It is highly desirable that workflow system evaluates security rules within a satisfactory (low) complexity. Such evaluation

complexity is not entirely considered by our preferred open source PEP engine, Balana. Therefore, we overlooked such need. However, it can penalize our overall system performance. Immaturity of such available PEP engines to fully implement all features of recently proposed XACML specification is another constraint of our proposed architecture.

We explore on-premise deployment of our proposed design, where the latency between PEP to PDP and attribute retrieval is minimal. In GPWFMS, PEP is placed near the resource and embedded within the same process as the services so that it improves the overall system performance. Processing complexity is one of the trade-offs while choosing security over performance. As business requirements increase and scale, the complex nature of computation and storage increases with the resulting large number of low-level access control rules. Also, there is a need for regular maintenance and audit of XACML policies, which can be difficult over time.

Our approach assumes that the communication channel is secure. One constraint is that access to web services need to be secured using authentication steps allowing only legitimate access requests. Such communication between front end client and the RESTful services is considered protected and secure using a secured protocol such as Transport Layer Security (TLS) or Secure Sockets Layer (SSL). Hence, the request and response communication cannot be intercepted by any attackers as well as sensitive attribute values are hidden or encrypted from them. Apparently, security of web service is another issue that is ignored in this work. Many secure authentication mechanisms can be implemented to make sure the only legit user can access the open web services. This authentication approach adds an extra dimension to the security of overall system architecture.

Due to the limited time constraints, we are unable to test all random scenarios and measure load and performance throughput of the system based on other policy rules like delegation and dynamic rules. However, our testing results indicate that the overall policy formation and handling used by our system is done correctly and can be generalized to any additional rules.

CHAPTER SIX: CONCLUSION

This study presented a complete conceptual framework of secure software design model with a viable implementation, which is missing in many existing literature works. Moreover, the API-driven reusable components are combined with context awareness to accommodate dynamic access policy enforcement. Also, it supports sophisticated features like Obligations and Delegation. We formulate conditions where such intricate features are desirable and have discussed the way to achieve these criteria in the context of the XACML architecture using ABAC. We propose a new reference software design model for ABAC based systems with obligations and delegation of authority rights. The proposed novel design allows externalization of authorization from code-level and provides secure abstracted services. This model also describes how the associated obligations (pre/post) with each action are enforced based on access control rules and how different users handle the dissemination of authority. So, using the proposed software design, we can solve the challenges such as automation and security managements alongside we can seamlessly integrate different access control constraints to make it more secure and robust.

The successful development, implementation, and validation of Grant Proposal Workflow Management System act as a proof-of-concept to the proposed software security architecture which is equally applicable to any other domain. Our strategy integrates secure architecture and design practices in the software development lifecycle to protect the overall application. The testing results prove that the proposed design model is a simple

yet general technique to specify and enforce fine-grained access control to maintain data integrity and confidentiality. Hence, the proposed software architecture applies to any workflow system that involves a group of people and their associated privileges.

Supporting scalability is considered as future work for this research. The advanced workflow system on top of the proposed model consists of many RESTful services that can be accessed by many users simultaneously. Rigorous testing of web services to handle multiple parallel requests is not conducted. Such tests can help to find out bottleneck and can provide a path for improving the performance. The level of security of the proposed model depends on the correctness of the written policies. Hence, accuracy and reliability of the written access control policies are a critical consideration. The manual task of defining and forming access control policy by security administrator is a cumbersome and tedious task. Due to the manual intervention of human factor, the policy definition and formulation process may lead to inconsistencies and errors that can cause a severe security risk to the overall system. This risk can be minimized by providing a level of automation and correction checking for access control policies. Our architectural model allows the system to contain a complete and non-repeating set of rules. In our delegation model, we have restricted some of the advanced delegation features like grant delegation, chained delegation, and multi-step delegation due to processing complexity. In future, such sophisticated delegation features can be explored and implemented within our proposed model.

REFERENCES

[1] S. Chaari, F. Biennier, C. Ben Amar, and J. Favrel, “An authorization and access control model for workflow,” *First Int. Symp. Control. Commun. Signal Process. 2004.*, no. April, pp. 141–148, 2004.

[2] E. Bertino, E. Ferrari, and V. Atluri, “The specification and enforcement of authorization constraints in workflow management systems,” *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 1, pp. 65–104, 1999.

[3] D. Georgakopoulos, M. Hornick, and A. Sheth, “An overview of workflow management: From process modeling to workflow automation infrastructure,” *Distrib. Parallel Databases*, vol. 3, no. 2, pp. 119–153, 1995.

[4] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-Based Access Control Models,” *IEEE Comput.*, vol. 29, no. 2, pp. 38–47, 1995.

[5] D. Hollingsworth, “Glossary, Terminology and Glossary, 3rd Edition. Document No WFMC-TC-1011. Workflow Management Coalition. Winchester, 1999,” *ReVision*, 1999.

[6] W. Huang and L. Wang, “Research of TRBAC model and the application in library management,” in *Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on*, 2010, pp. 337–339.

[7] S. Lakkaraju and D. Xu, “Integrated Modeling and Analysis of Attribute Based Access Control Policies and Workflows in Healthcare,” *2014 Int. Conf. Trust. Syst. their Appl.*, pp. 36–43, 2014.

[8] L. Sainan, “Task-role-based access control model and its implementation,” *2010 2nd Int. Conf. Educ. Technol. Comput.*, pp. V3-293-V3-296, 2010.

[9] Y. Liu, K. Xu, and J. Song, “A task-attribute-based workflow access control model,” Proc. - 2013 IEEE Int. Conf. Green Comput. Commun. IEEE Internet Things IEEE Cyber, Phys. Soc. Comput. GreenCom-iThings-CPSCom 2013, pp. 1330–1334, 2013.

[10] V. C. Hu, K. Scarfone, and R. Kuhn, NIST Special Publication 800-162 DRAFT - FINAL Guide to Attribute Based Access Control (ABAC) Definition and Considerations NIST Special Publication 800-162 DRAFT - FINAL Guide to Attribute Based Access Control (ABAC) Definition and Considerations. 2013.

[11] W. Tolone, G.-J. Ahn, T. Pai, and S.-P. Hong, “Access control in collaborative systems,” *ACM Comput. Surv.*, vol. 37, no. 1, pp. 29–41, 2005.

[12] Y. Lu and L. Zhang, “Domain administration of task-role based access control for process collaboration environments,” in *5th International Conference on Information Assurance and Security, IAS 2009*, 2009, vol. 1, pp. 643–647.

[13] J. Zhang, J. Sun, N. Li, and C. Hu, “A conditioned secure access control model on multi-weighted roles in workflow system,” in *2005 International Conference on Control and Automation*, 2005, vol. 2, p. 1068–1073 Vol. 2.

[14] N. Damianou, N. Dulay, and E. Lupu, “The ponder policy specification language,” *Proc. Policy 2001 Work. Policies Distrib. Syst. Networks*, pp. 18–39, 2001.

[15] H. Movahednejad, S. Bin Ibrahim, M. Sharifi, H. Bin Selamat, and S. G. H. Tabatabaei, “Security-aware web service composition approaches,” *Proc. 13th Int. Conf. Inf. Integr. Web-based Appl. Serv. - iiWAS ’11*, p. 112, 2011.

[16] E. Yuan and J. Tong, “Attributed Based Access Control (ABAC) for web services,” *Proc. - 2005 IEEE Int. Conf. Web Serv. ICWS 2005*, vol. 2005, pp. 561–569, 2005.

[17] T. Priebe, W. Dobmeier, and N. Kamprath, “Supporting attribute-based access control with ontologies,” *Proc. - First Int. Conf. Availability, Reliab. Secur. ARES 2006*, vol. 2006, pp. 465–472, 2006.

[18] H. Shen and F. Hong, “An Attribute-Based Access Control Model for Web Services,” *Proc. Seventh IEEE Int. Conf. Parallel Distrib. Comput. Appl. Technol.*, pp. 74–79, 2006.

[19] R. Lepro, “Cardea : Dynamic Access Control in Distributed Systems,” *NAS Tech. Rep. NAS-03-020*, pp. 1–31, 2003.

[20] J. Herrmann, “Access Control in OGC Web Service based Architectures Categories and Subject Descriptors,” *Access*, pp. 60–67, 2011.

[21] R. Abassi, F. Jacquemard, M. Rusinowitch, and S. G. El Fatmi, “XML access control: From XACML to annotated schemas,” *2010 2nd Int. Conf. Commun. Networking, ComNet 2010*, no. October 2015, 2010.

[22] M. Lischka, “Dynamic obligation specification and negotiation,” *Proc. 2010 IEEE/IFIP Netw. Oper. Manag. Symp. NOMS 2010*, pp. 155–162, 2010.

[23] H. Jebbaoui, A. Mourad, H. Otrok, and R. Haraty, “Semantics-based approach for detecting flaws, conflicts and redundancies in XACML policies,” *Comput. Electr. Eng.*, vol. 44, pp. 91–103, 2015.

[24] U. M. Mbanaso, G. S. Cooper, D. W. Chadwick, and A. Anderson, “Obligations of trust for privacy and confidentiality in distributed transactions,” *Internet Res.*, vol. 19, no. 2, p. 153, 2007.

[25] T. Sans, F. Cuppens, and N. Cuppens-Boulahia, “A framework to enforce access control, usage control and obligations,” *Ann. des Telecommun. Telecommun.*, vol. 62, no. 11–12, pp. 1329–1352, 2007.

[26] Y. Elrakaiby, F. Cuppens, and N. Cuppens-Boulahia, “Formal enforcement and management of obligation policies,” *Data Knowl. Eng.*, vol. 71, no. 1, pp. 127–147, 2012.

[27] D. W. Chadwick, S. Otenko, and T. A. Nguyen, Adding support to XACML for multi-domain user to user dynamic delegation of authority, vol. 8, no. 2. 2009.

[28] A. . Fallis, “Communications and Multimedia Security,” *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2013.

[29] L. X. Feng and F. D. Guo, “Composing administrative scope of delegation policies based on extended XACML,” *Proc. - IEEE Int. Enterp. Distrib. Object Comput. Work. EDOC*, pp. 467–470, 2006.

[30] S. Fischer-Hübner, C. Lambrinoudakis, and J. Lopez, “Trust, Privacy and Security in Digital Business: 12th International Conference, TrustBus 2015 Valencia, Spain, September 1-2, 2015 Proceedings,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9264, no. October, p. 5, 2015.

[31] C. Ye, Z. Wu, and Y. Fu, “An attribute-based delegation model and its extension,” *J. Res. Pract. Inf. Technol.*, vol. 38, no. 1, pp. 3–16, 2006.

[32] D. W. Chadwick and K. Fatema, “A privacy preserving authorisation system for the cloud,” in *Journal of Computer and System Sciences*, 2012, vol. 78, no. 5, pp. 1359–1373.

[33] M. Tomaiuolo, “dDelega:,” *Int. J. Inf. Secur. Priv.*, vol. 7, no. 3, pp. 53–67, 2013.

[34] V. Atluri and J. Warner, “Supporting Conditional Delegation in Secure Workflow Management Systems,” *Proc. Tenth ACM Symp. Access Control Model. Technol. - SACMAT ’05*, p. 49, 2005.

[35] J. H. Saltzer and M. D. Schroeder, “The Protection of Information in Computer Systems,” *Proc. IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[36] B. Nuseibeh and S. Easterbrook, “Requirements Engineering: A Roadmap,” in *Proceedings of the Conference on The Future of Software Engineering*, 2000, pp. 35–46.

[37] P. T. Devanbu and S. Stubblebine, “Software Engineering for Security: A Roadmap,” in *Proceedings of the Conference on The Future of Software Engineering*, 2000, pp. 227–239.

[38] K. Hasebe, M. Mabuchi, and A. Matsushita, “Capability-based delegation model in RBAC,” *Proceeding 15th ACM Symp. Access Control Model. Technol. - SACMAT ’10*, p. 109, 2010.

[39] E. Barka and R. Sandhu, “Framework for role-based delegation models,” *Proc. - Annu. Comput. Secur. Appl. Conf. ACSAC*, vol. 2000–Janua, pp. 168–176, 2000.

[40] A. Ali, U. Habiba, and M. A. Shibli, “Taxonomy of Delegation Model,” *2015 12th Int. Conf. Inf. Technol. - New Gener.*, pp. 218–223, 2015.

[41] J. Crampton and H. Khambhammettu, “Delegation in role-based access control,” in *International Journal of Information Security*, 2008, vol. 7, no. 2, pp. 123–136.

[42] L. Bussard, A. Nano, and U. Pinsdorf, “Delegation of access rights in multi-domain service compositions,” *Identity Inf. Soc.*, vol. 2, no. 2, pp. 137–154, 2009.

[43] Z. Malik and A. Bouguettaya, *Trust Management for Service-Oriented Environments*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[44] Oasis, “eXtensible Access Control Markup Language,” *OASIS Stand.*, no. February, p. 141, 2005.

APPENDIX A

Use Case Descriptions for GPWFMS

1. Add/Create proposal:

This use case represents the process of adding/creating a proposal by Tenured/Non-Tenured Faculty.

Use case #	UC-1
Use case name	Create/Add proposal.
Actor	Principal Investigator (PI)
Goal	To create a new proposal.
Preconditions	1. The actor has an account on the system. 2. The actor job position should be Tenured/Non-Tenured track Faculty.
Main Flow	1. The actor login to the system. 2. The actor selects the "Add new Proposal" action. 3. The system receives the actor request and redirects the user to the new proposal page. 4. The actor fills the "Investigator Information" by filling the Co-PI and Senior Personal by selecting the "Add Co-PI" action and "Add Senior Personnel" action. 5. The actor fills the "Project Information" section. The actor fills the "Project Title, Project Type, Due Date, Project Period: From, TO: Type of Request, and Location of Project" fields. 6. The actor fills the "Sponsor and Budget Information" by filling: "Name of Granting Agency, Direct Costs, Total Costs, F&A Costs, and F&A Rate" fields. 7. The actor fills "*Cost Share Information*" by filling: "*Is Institutional committed cost share included in the proposal? And Is Third Party committed cost share included in the proposal?*" fields. 8. The actor fills the "*University Commitments*" by filling: "*Will new or renovated space/facilities be required? Will rental space be*

	required? and Does this project require institutional commitments beyond the end date of the project?" fields. 9. The actor fills the "*Conflict of Interest and Commitment Information*" section by filling: "*Is there a financial conflict of interest related to this proposal? Has the financial conflict been disclosed? and Has there been a material change to your annual disclosure form?"* fields. 10. The actor fills the "*Compliance Information*" section by filling: "*Does this project involve the use of Human Subjects? Does this project involve the use of Vertebrate Animals? Does this project include Biosafety concerns? and Does this project have Environmental Health & Safety concerns?*" fields. 11. The actor fills the "*Additional Information*" section by filling: "*Do you anticipate payment(s) to foreign nationals or on behalf of foreign nationals? Do you anticipate course release time? and Are the proposed activities related to Center for Advanced Energy Studies?*" fields. 12. The actor fills the "*Collaboration Information*" section by filling: *Does this project involve Non-funded collaborations?*" filed. 13. The actor fills the "*Proprietary/Confidential Information*" section by filling: "*Does this proposal contain any confidential information which is Proprietary that should not be publicly released? Will this project involve intellectual property in which the University may own or have an interest?*" fields. 14. The actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 15. The actor fills "*Appendices*" section by using the file upload action. 16. The actor selects the save action to keep the proposal. 17. The system sends notifications to the Co-PI(s) and senior personal. 18. The system records the request in the user audit log

Post-Condition	1. The system saves the proposal with correct data submitted by the actor. 2. The actor can access the proposal.
Alternative flow	NONE
Exception flow	NONE
Recovery flow	NONE

2. Delete proposal by principal investigator (PI) use case:

This use case represents the process of deleting proposal by PI.

Use case #	UC-2
Use case name	Delete a proposal by PI
Actor	Principal Investigator (PI)
Goal	To delete the proposal document.
Preconditions	1. The proposal not submitted by PI.
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor selects the "Delete" action. 4. The system processes the requests and deletes the proposal. 5. The system sends a confirmation message. 6. The system sends notification PI, Co-PI, Senior Personnel. 7. The system records that in user audit log 8. The system records that in the system log file.
Post-Condition	1. The system successfully deletes the proposal sheet. 2. The actor cannot find, open, and/or edit the proposal.
Alternative flow	2.a **The actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The Actor Deletes proposal in MF.*
Exception flow	NONE
Recovery flow	NONE

3. Co-PI signs and updates the proposal.

This use case represents the process of Signing the proposal by Co-PI.

Use case #	UC-3
Use case name	Co-PI signs and updates the proposal.
Actor	Co-PI
Goal	Co-PI signs and updates the proposal.
Preconditions	1. The Co-PI is added to the proposal by PI.
Main Flow	1. The actor login to his/her account. 2. The actor select the proposal by selecting edit action. 3. The actor can update "Investigator Information" section in the proposal. 4. The actor signs the proposal by filling the signature, date, and note fields. 5. The system updates the proposal status and saves it. 6. The system sends a confirmation message. 7. The system records that on the user audit log. 8. The system records in the system log.
Post-Condition	1. The proposal status changed to ready to submit by PI.
Alternative flow	2.a **The actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The actor signs the proposal in MF.*
Exception flow	NONE.
Recovery flow	NONE.

4. Submission proposal by principal investigator (PI) use case description:

This use case represents the process of submission of a proposal by PI to Department Chair.

Use case #	UC-4
Use case name	Submit proposal by principal investigator (PI).

Actor	Principal Investigator (PI)
Goal	To submit the proposal to the department chair.
Preconditions	1. The PI created the proposal and signed it. 2. The Co-PI(s) signed the proposal. 3. The proposal status not submitted.
Main Flow	1. The actor login to account. 2. The actor selects "My proposals" action. 3. The actor selects the proposal by selecting the edit proposal action. 4. The system opens the proposal in edit mode. 5. The actor signs the proposal. 6. The actor selects the submit action. 7. The system sends a notification to the department chair, PI, Co-PI(s) and Senior Personnel. 8. The system records the request in the user audit log.
Post-Condition	1. The proposal Status changed to waiting for chair approval. 2. The actor has read access to the proposal.
Alternative flow	2.a **The actor uses the research engine** 2.a.1 The actor inserts the proposal information in the search fields. 2.a.2 The system returns the search result. 2.a.3 The actor selects the proposal. The use case continuous at *The actor selects the submit action in MF*
Exception flow	**4.a Co-PI(s) not signed the proposal** 4.a.1 The system shows an error message that Co-PIs are not signed on the proposal.
Recovery flow	NONE

5. Department Chair approve/disapprove the proposal.

This use case represents the process of approving and disapproving a proposal by Department Chair.

Use case #	UC-5

Use case name	Department Chair approve/disapprove the proposal.
Actors	Department Chair
Goal	Department Chair approve/disapprove proposal.
Preconditions	1. The proposal is signed by all Co-PI. 2. The proposal is signed by the PI. 3. The proposal is submitted by PI. 4. The proposal status is ready for Chair approval.
Main Flow	1. The actor logged into the system. 2. The actor selects “My proposal” action. 3. The actor selects the proposal by selecting “edit” action. 4. The actor signs the proposal by filling the signature, date, and note fields. 5. The actor approves or disapproves the proposal by selecting approve/disapprove action. 6. If the actor approves the proposal, the system will send notifications to the PI, Co-PI, IRB and University Business Manager, Else, the system will send a notification to system sends an email to PI, Co-PI, and all Department Chairs. 7. The system updates the proposal status and saves it. 8. The system sends a confirmation message. 9. The system records that on the user audit log. 10. The system records in the system log.
Post-condition	1. If the actor approved the proposal, the proposal status will change to Ready for Business Manager Approval and/or IRB. 2. If the actor disapproves the proposal, the proposal status will change to not submitted.

Alternative Flow	**2.a The actor uses the notification tab to select the proposal** 2.a.1 The actor checks and opens the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The actor approves/disapproves the proposal in MF*.
Exception flow	NONE
Recovery flow	NONE

6. Department Chair Delegates Associate Chair.

This use case represents the process of Delegation of Authority by Department Chair to Associate Chair of his/her own Department

Use case #	UC-6
Use case name	Department Chair Delegates Associate Chair of his/her own Department.
Actors	Department Chair
Goal	Department Chair Delegates Associate Chair of his/her own Department.
Preconditions	1. The actor has an account on the system. 2. The actor's position title must be Department Chair.
Main Flow	1. The actor logged into the system. 2. The actor selects "Delegation" menu. 3. The actor selects "Add New Delegation" action. 4. The system receives the actor request and redirects the user to the new delegation page. 5. The actor selects the "Delegate To" that is bind to delegate users based on policy rule specified such as User with position title of Associate Chair from his/her own Department. 6. The actor selects the "Delegate Actions" by selecting multiple checkboxes based on a policy defined for current actor's context.

	7. The actor selects the range of temporal delegation period using starting date and ending date for the delegation. 8. The actor fills the reason for the current delegation. 9. The actor saves the delegation information. 10. The system sends notifications to the selected delegatee and current delegator. 11. The system records that on the delegation audit log.
Post-condition	1. The system saves the delegation with correct data submitted by the actor. 2. The actor can access the delegation for edit and revocation.
Alternative Flow	NONE
Exception Flow	NONE
Recovery Flow	NONE

7. Department Chair revokes delegation from Associate Chair.

This use case represents the process of Revocation of Delegation of Authority by Department Chair from his/her Delegatee.

Use case #	UC-7
Use case name	Department Chair Revokes Delegation of Authority from Associate Chair of his/her own Department.
Actors	Department Chair
Goal	Department Chair Revokes Delegation of Authority from Associate Chair of his/her own Department.

Preconditions	1. The actor has an account on the system. 2. The actor's position title must be Department Chair. 3. The actor must have existing Delegation.
Main Flow	1. The actor logged into the system. 2. The actor selects "Delegation" menu. 3. The actor chooses a specific delegation by selecting the Edit delegation action. 4. The system opens the selected delegation in edit mode. 5. The actor chooses the Revoke action. 6. The system sends notifications to the chosen delegatee and current delegator. 7. The system records that on the delegation audit log.
Post-condition	1. The system saves the delegation with revocation status. 2. The actor cannot access the delegation for edit and revocation again.
Alternative Flow	**3.a The actor uses the Revoke action to revoke the delegation** 3.a.1 The actor selects a specific delegation. 3. a.2 The actor selects and confirms the Revoke delegation action. The use case continuous at *The actor revokes the delegatee in MF.*
Exception Flow	NONE
Recovery Flow	NONE

8. Business Manager approves/disapproves the proposal.

This use case represents the process of approving and disapproving a proposal by Business Manager.

Use case #	UC-8
Use case name	Business Manager approve/disapprove proposal.

Actors	Business Manager
Goal	Business Manager Approve/Disapprove the proposal.
Preconditions	1. The proposal signed by all Department Chair. 2. The proposal approved by all Department Chair. 3. The proposal status is ready for Business Manager approval.
Main Flow	1. The actor is logged in. 2. The actor selects “My proposal” action. 3. The actor selects the proposal by selecting “edit” action. 4. The actor can edit the “Sponsor and Budget Information” section in the proposal. 5. The actor signs the proposal by filling the signature, date, and note fields. 6. The actor approves or disapproves the proposal by selecting approve/disapprove action. 7. If the actor approves the proposal, the system will send notifications to the PI, Co-PI, IRB and the Dean, Else, the system will send a notification to system sends an email to PI, Co-PI, Department Chair, IRB, and all Business Managers. 8. The system updates the proposal status and saves it. 9. The system sends a confirmation message. 10. The system records that on the user audit log. 11. The system records in the system log.
Post-condition	1. If the actor approved the proposal, the proposal status would change to ready for Dean’s approval. 2. If the actor disapproves the proposal, the proposal status will change to not submitted.
Alternative Flow	2.a **The actor uses the notification tab to select the proposal** 2.a.1 The actor checks and opens the notification tab

	2.a.2 The actor selects the proposal. The use case continuous at *The actor approves/disapproves the proposal in MF*.
Exception Flow	NONE
Recovery Flow	NONE

9. Approve/Disapprove Proposal by Dean use case:

This use case represents the process of approving and disapproving a proposal by Dean.

Use case #	UC-9
Use case name	Approve/Disapprove proposal by dean.
Actor	Dean
Goal	To approve/disapprove the proposal.
Preconditions	1. The proposal signed by all Business Manager. 2. The proposal approved by all Business Manager. 3. The proposal status is ready for Dean approval.
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor signs the proposal by filling the "Signature, Date and Note" fields. 4. The actor approves or disapproves the proposal by selecting approve/disapprove action. 5. **If** the actor approves the proposal, the system will send notifications to the PI, Co-PI, Senior Personnel and University Research Administrator, **Else**, the system will send a notification to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB. 6. The system updates the proposal status and saves it. 7. The system sends a confirmation message. 8. The system records that on the user audit log.

Post-Condition	1. If the actor approved the proposal, and the IRBs approved the proposal status changed to the ready for Research administrator approval else the status will stay ready for IRB approval. 2. If the actor disapproves the proposal, the proposal status changed to not submitted, and, clear all signatures.
Alternative flow	2.a **The uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The actor approves/disapproves the proposal in MF.*
Exception flow	NONE.
Recovery flow	NONE.

10. IRB approve/disapprove the proposal.

This use case represents the process of approving and disapproving a proposal by IRB.

Use case #	UC-10
Use case name	IRB approve/disapprove proposal.
Actors	IRB
Goal	Business Manager approve/disapprove proposal.
Preconditions	1. The proposal status is ready for IRB approval. 2. The proposal has a compliance
Main Flow	1. The actor is logged in. 2. The actor selects “My proposal” action. 3. The actor selects the proposal by selecting “edit” action. 4. The actor signs the proposal by filling the signature, date, and note fields. 5. The actor approves or disapproves the proposal.

	6. If the actor approves the proposal, the system will send notifications to the PI, Co-PI, Senior Personnel and Research Administrator, Else, the system will send a notification to system sends an email to PI, Co-PI, and all Department chair. 7. The system updates the proposal status and saves it. 8. The system sends a confirmation message. 9. The system records that on the user audit log.
Post-condition	1. If the actor approved the proposal and the Deans approved, the proposal status will change to the ready for Research Administrator's approval else will remain ready for Dean approval. 2. If the actor disapproves the proposal, the proposal status will change to not submitted.
Alternative Flow	2.a **The actor uses the notification tab to select the proposal** 2.a.1 The actor checks and opens the notification tab. 2.a.2 The actor selects the proposal. The use case continuous at The actor approves/disapproves the proposal in MF.
Exception Flow	NONE.
Recovery Flow	NONE.

11. Approve/Disapprove proposal by Research Administrator.

This use case represents the process of approving and disapproving a proposal by Research Administrator.

Use case #	UC-11
Use case name	Approve/Disapprove proposal by Research Administrator.
Actor	Research Administrator
Goal	To approve/disapprove the proposal.
Preconditions	1. The proposal status is ready for Research Administrator.

Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor signs the proposal by filling the signature, date, and note fields. 4. The actor can update the following sections of the proposal, such as "Investigator Information", "Project Information", Sponsor and Budget Information", "Cost Share Information", "University Commitments", "Conflict of Interest and Commitment Information", "Compliance Information", "Additional Information", "Collaboration Information", "Proprietary/Confidential Information", "Certification/Signatures", and "OSP Section". 5. The actor approves or disapproves the proposal. 6. **If** the actor approves the proposal, the system will send notifications to the PI, Co-PI, Senior Personnel and University Research Director, Else, the system will send a notification to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB. 7. The system updates the proposal status and saves it. 8. The system sends a confirmation message. 9. The system records that on the user audit log. 10. The system records in the system log.
Post-Condition	1. If the actor approved the proposal, the proposal status changed to ready for Research Director approval. 2. If the actor disapproves the proposal, the proposal status changed to not submitted, and, clear all signatures.
Alternative flow	2.a **The actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at <u>The actor approve/disapprove the proposal in MF.</u>
Exception flow	NONE.

Recovery flow	NONE.

12. Withdraw the proposal by Research Administrator.

This use case represents the process of withdrawing a proposal by Research Administrator.

Use case #	UC-12
Use case name	Withdraw proposal by Research Administrator.
Actor	Research Administrator
Goal	To withdraw the proposal.
Preconditions	1. The proposal status ready for research administrator approval.
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor signs the proposal by filling the "Signature, Date and Note" fields. 4. The actor withdraws a proposal by selecting the withdraw action. 5. The system updates the proposal status and saves it. 6. The system sends the confirmation message. 7. The system sends a notification to the PI, Co-PI, Senior Personnel Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB The system records that on the user audit log. 8. The system records the request in the user audit log. 9. The system records in the system log.
Post-Condition	1. The proposal status changed to withdrawn. 2. The proposal cannot be updated by PI.
Alternative flow	2.a **The actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The actor withdraw proposal in MF.*
Exception flow	NONE.

Recovery flow	NONE.

13. Approve/Disapprove proposal by Research Director:

This use case represents the process of approving and disapproving a proposal by Research Director.

Use case #	UC-13
Use case name	Approve/Disapprove proposal by Research Director.
Actor	Research Director
Goal	To approve/disapprove the proposal.
Preconditions	1. The proposals signed by all research administrators. 2. The proposal approved by all research administrators. 3. The proposal status is ready for Research Director approval.
Main Flow	1. The actor login to his/her account. 2. The actor selects “My proposal” action. 3. The actor selects the proposal by selecting edit action. 4. The actor can update the “OSP” section fields in the proposal. 5. The actor signs the proposal by filling the “Signature, Date and Note” fields. 6. The actor approves or disapproves the proposal by selecting the approve/disapprove action. 7. If the actor approves the proposal, the system will send notifications to the PI, Co-PI, Senior Personnel, University Research Administrator, Else, the system will send a notification to System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, IRB and all Deans. 8. The system updates the proposal status and saves it. 9. The system sends a confirmation message. 10. The system records that on the user audit log.
Post-Condition	1. If the actor approved the proposal, the proposal status changed to ready for search administrator submission.

	2. If the actor disapproves the proposal, the proposal status changed to not submitted, and, clear all signatures.
Alternative flow	2.a **The actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The actor approve/disapprove the proposal in MF*.
Exception flow	NONE.
Recovery flow	NONE.

14. Delete proposal by Research Director use case:

This use case represents the process of deleting proposal by Research Director.

Use case #	UC-14
Use case name	Delete proposal by Research Director
Actor	Research Director
Goal	To delete the proposal.
Preconditions	1. The proposal status is ready for Research Director Approval.
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor selects the "Delete" action. 4. The system processes the requests and deletes the proposal. 5. The system sends a confirmation message. 6. The system sends a notification to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB. 7. The system records that in user audit log 8. The system records that in the system log file.
Post-Condition	1. The system successfully deletes the proposal sheet. 2. The proposal status will change to deleted. 3. The PI cannot updates/edits the proposal. 4. The PI cannot be submitted again.

Alternative flow	2.a **The actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The actor Delete proposal in MF*.
Exception flow	NONE
Recovery flow	NONE

15. Submit the proposal by Research Administrator:

This use case represents the process of submitting a proposal by Research Administrator.

Use case #	UC-15
Use case name	Submit proposal to research administrator.
Actor	Research Administrator
Goal	To submit the proposal.
Preconditions	1. The proposal approved by all research directories. 2. The proposals status ready for research administrator submission
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor can update the following sections of the proposal, such as "Investigator Information", "Project Information", Sponsor and Budget Information", "Cost Share Information", "University Commitments", "Conflict of Interest and Commitment Information", "Compliance Information", "Additional Information", "Collaboration Information", "Proprietary/Confidential Information", "Certification/Signatures", and "OSP Section". 4. The actor signs the proposal by filling the "Signature, Date and Note" fields. 5. The actor submits a proposal. 6. The system updates the proposal status and saves it. 7. The system sends the confirmation message.

	8. The system sends a notification to the PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Director and IRB. 9. The system records request on the user audit log. 10. The system records request on the system log.
Post-Condition	1. The proposal status changed to be submitted by research administrator.
Alternative flow	2.a **The actor uses the notification tab to select the proposal** 2.a.1 The actor selects the notification tab. 2.a.2 The actor selects the proposal. The use case continuous at *The actor signs the proposal in MF.*
Exception flow	NONE.
Recovery flow	NONE.

16. Archive proposal by Research Director.

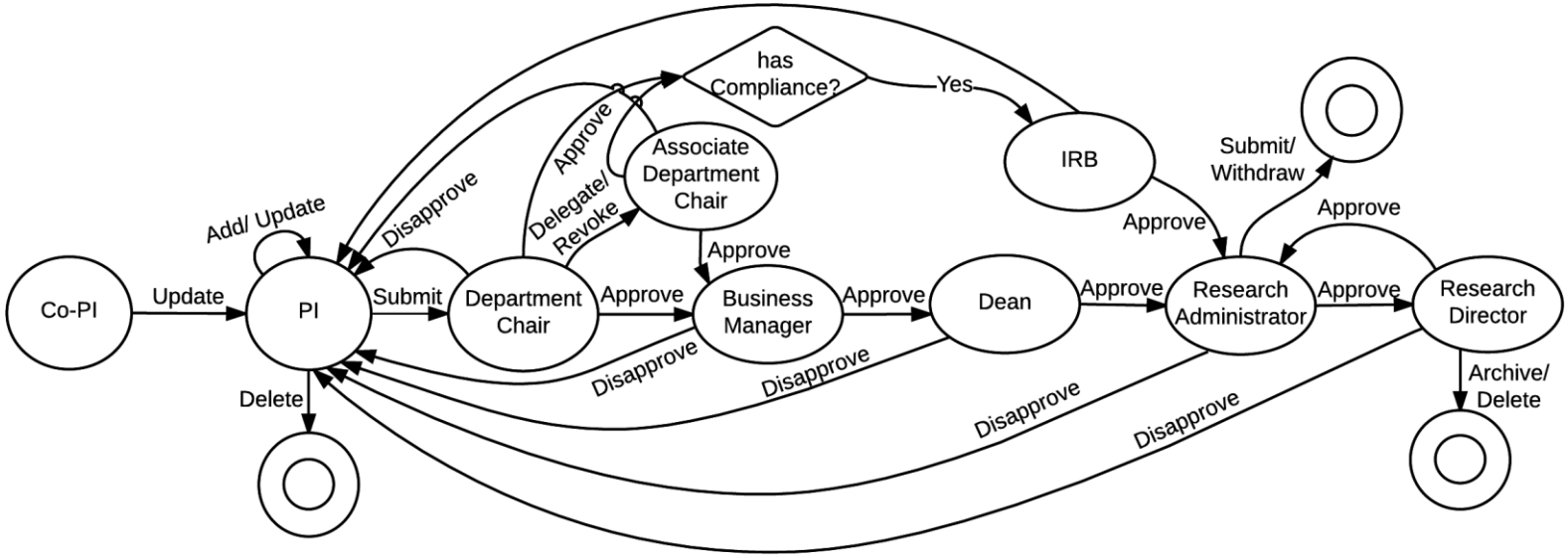
This use case represents the process of the archiving proposal by Research Director.

Use case #	UC-16
Use case name	Archive proposal.
Actor	Research Director
Goal	To archive the proposal.
Preconditions	1. The proposal approved by Research Administrator.
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor selects the "Archive" action. 4. The system processes the requests and archives the proposal. 5. The system sends a confirmation message. 6. The system sends a notification to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB. 7. The system records that in user audit log.

	8. The system records that in the system log file.
Post-Condition	1. The proposal status changed to archived 2. The proposal cannot be updated by any actor.
Alternative flow	2.a **The actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The actor selects Archive proposal in MF*.
Exception flow	NONE.
Recovery flow	NONE.

APPENDIX B

State Diagram of GPWFMS with Delegation



APPENDIX C

Attribute Metadata Definition

Attribute	Category	Type	Value
SubmittedByPI	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	SUBMITTED, NOTSUBMITTED
ReadyForSubmissionByPI	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	True, False
DeletedByPI	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	DELETED, NOTDELETED
ApprovedByDepartmentChair	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	APPROVED, DISAPPROVED, READYFORAPPROVAL, NOTREADYFORAPPROVAL
ApprovedByBusinessManager	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	APPROVED, DISAPPROVED, READYFORAPPROVAL, NOTREADYFORAPPROVAL
ApprovedByIRB	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	APPROVED, DISAPPROVED, READYFORAPPROVAL, NOTREADYFORAPPROVAL

ApprovedByDean	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	APPROVED, DISAPPROVED, READYFORAPPROVAL, NOTREADYFORAPPROVAL
ApprovedByUniversityResearchAdministrator	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	APPROVED, DISAPPROVED, READYFORAPPROVAL, NOTREADYFORAPPROVAL
WithdrawnByUniversityResearchAdministrator	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	WITHDRAWN, NOTWITHDRAWN
ApprovedByUniversityResearchDirector	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	APPROVED, DISAPPROVED, READYFORAPPROVAL, NOTREADYFORAPPROVAL
DeletedByUniversityResearchDirector	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	DELETED, NOTDELETED
SubmittedByUniversityResearchAdministrator	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	SUBMITTED, NOTSUBMITTED
ArchivedByUniversityResearchDirector	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	ARCHIVED, NOTARCHIVED
position.type	urn:oasis:names:tc:xacml:1.0:subject-category:access-subject	http://www.w3.org/2001/XMLSchema#string	Tenured/Tenured-track Faculty, Non-Tenured-track research Faculty, Teaching Faculty, Research staff, Professional staff, Administrator

position.title	urn:oasis:names:tc:xacml:1.0:subject-category:access-subject	http://www.w3.org/2001/XMLSchema#string	Distinguished Professor, Professor, Associate Professor, Assistant Professor, Research Professor, Associate Research Professor, Assistant Research Professor, Clinical Professor, Clinical Associate Professor, Clinical Assistant Professor, Visiting Professor, Visiting Associate Professor, Visiting Assistant Professor, Lecturer, Senior Lecturer, Adjunct Professor, Research Associate, Research Scientist, Senior Research Scientist, IRB, Business Manager, University Research Administrator, Department Administrative Assistant, Department Chair, Associate Chair, Dean, Associate Dean, Research Administrator, University Research Director
proposal.role	urn:oasis:names:tc:xacml:1.0:subject-category:access-subject	http://www.w3.org/2001/XMLSchema#string	PI, Co-PI, Senior Personnel
proposal.section	urn:oasis:names:tc:xacml:1.0:attribute-category:resource	http://www.w3.org/2001/XMLSchema#string	Whole Proposal, Investigator Information, InvestigatorInformation.PI, InvestigatorInformation.Co-PI, InvestigatorInformation.Senior-Personnel, Project

			Information, Sponsor and Budget Information, Cost Share Information, University Commitments, Conflict of Interest and Commitment Information, Compliance Information, Additional Information, Collaboration Information, Proprietary/Confidential Information, Certification/Signatures, OSP Section, Appendices, Audit Log
proposal.action	urn:oasis:names:tc:xacml:1.0:attribute-category:action	http://www.w3.org/2001/XMLSchema#string	Add, Add Co-PI, Add Senior Personnel, Save, Submit, Approve, Disapprove, Withdraw, Archive, Delete, View, Edit
device.type	urn:oasis:names:tc:xacml:1.0:subject-category:environment	http://www.w3.org/2001/XMLSchema#string	Android Device, Windows Device, iOS Device
network.type	urn:oasis:names:tc:xacml:1.0:subject-category:environment	http://www.w3.org/2001/XMLSchema#string	Campus, Outside Campus
department	urn:oasis:names:tc:xacml:1.0:subject-category:access-subject	http://www.w3.org/2001/XMLSchema#string	Computer Science, Electrical Engineering, Computer Engineering, Physics, Chemistry

APPENDIX D

Functional and Access Control Requirements

Action: Add					
Rule	**Action**	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligations**
1. AddProposalByFaculty-Rule1	Add A New Proposal by Tenured/Tenured-track faculty (Permit)	position.type = Tenured/Tenured-track faculty proposal.section = Whole Proposal proposal.action = Add			
	Add A New Proposal by Non-Tenured-track research faculty (Permit)	position.type = Non-Tenured-track research faculty proposal.section = Whole Proposal proposal.action = Add			
2. CannotAddProposalByOtherStaff-Rule2	Cannot Add a New Proposal by other Staff (Deny)	position.type = <Teaching faculty || Research staff || Professional staff || Administrator> proposal.section = Whole Proposal proposal.action = Add			

Action: Add Co-PI					
Rule	**Action**	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligations**

3. AddCo-PIByPI-Rule3	Co-PI can be Added by PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED proposal.section = InvestigatorInformation.Co-PI proposal.action = Add Co-PI			
4. CannotAddCo-PIByCoPI-Rule4	Co-PI cannot be Added by Co-PI (Deny)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED ReadyForSubmissionByPI = False proposal.section = InvestigatorInformation.Co-PI proposal.action = Add Co-PI			

Action: Add Senior Personnel					
Rule	**Action**	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligations**
5. AddSeniorPersonnelByPI-Rule5	Senior Personnel can be Added by PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED proposal.section = InvestigatorInformation.Senior-Personnel proposal.action = Add Senior Personnel			
6. AddSeniorPersonnelByCoPI-Rule6	Senior Personnel can be Added by Co-PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED ReadyForSubmissionByPI = False			

		proposal.section = InvestigatorInformation.Senior-Personnel proposal.action = Add Senior Personnel			

Action: Save					
Rule	**Action**	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligations**
7. SaveProposalByFaculty-Rule7	Save a New Proposal by Tenured/Tenured-track faculty (Permit)	position.type = Tenured/Tenured-track faculty proposal.section = Whole Proposal proposal.action = Save			System sends an email to PI, Co-PI, Senior Personnel
	Save a New Proposal by Non-Tenured-track research faculty (Permit)	position.type = Non-Tenured-track research faculty proposal.section = Whole Proposal proposal.action = Save			System sends an email to PI, Co-PI, Senior Personnel
8. SaveProposalByPI-Rule8	Update an Existing Proposal by PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED proposal.role = PI proposal.section = Whole Proposal proposal.action = Save	If PI, Co-PIs have signed then ReadyForSubmissionByPI = True else		System sends an email to PI, Co-PI, Senior Personnel

			ReadyForSubmissionByPI = False		
9. SaveProposalByCo-PI-Rule9	Update Existing Proposal by Co-PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED ReadyForSubmissionByPI = False proposal.role = Co-PI proposal.section = Whole Proposal proposal.action = Save	If PI, Co-PIs have signed then ReadyForSubmissionByPI = True else ReadyForSubmissionByPI = False		System sends an email to PI, Co-PI, Senior Personnel

Action: Submit					
Rule	**Action**	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligations**
10. SubmitProposalByPI-Rule10a	Submit Proposal by PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED ReadyForSubmissionByPI = True proposal.role = PI proposal.action = Submit	If all PI, Co-PIs have signed then SubmittedByPI =	PI signs the proposal	System sends an email to PI, Co-PI, Senior Personnel,

		Condition: signedByAllCoPIs =true	SUBMITTED ApprovedBy DepartmentChair = READYFOR APPROVAL else ReadyForSubmissionByPI = False		Department Chair If signedByAllCoPIs = true
11. NotSubmitProposalByPI-Rule10b	Not Submit Proposal by PI (Deny)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED ReadyForSubmissionByPI = True proposal.role = PI proposal.action = Submit Condition: signedByAllCoPIs =false			
12. NotSubmitProposalByCoPI-Rule11	Not Submit Proposal by Co-PI (Deny)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED ReadyForSubmissionByPI = False proposal.role = Co-PI proposal.action = Submit			
12a.SubmitProposalByUniversityResearchAdministrator-Rule12a	Submit By University Research	SubmittedByUniversityResearchAdministrator = NOTSUBMITTED	SubmittedBy UniversityResearchAdmin	University Research Administrat	System sends an email to PI, Co-PI, Senior

	Administrator (Permit)	ApprovedByUniversityResearchDirector = APPROVED position.title = University Research Administrator proposal.action = Submit Condition: irbApprovalRequired =false	istrator = SUBMITTED	or signs the proposal	Personnel, Department Chair, Business Manager, Dean, University Research Administrator
12b.SubmitProposalByUniversityResearchAdministrator-Rule12b	Submit By University Research Administrator (Permit)	SubmittedByUniversityResearchAdministrator = NOTSUBMITTED ApprovedByUniversityResearchDirector = APPROVED position.title = University Research Administrator proposal.action = Submit Condition: irbApprovalRequired =true	SubmittedByUniversityResearchAdministrator = SUBMITTED	University Research Administrator signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Director and IRB

Action: Approve

Rule	Action	Pre-Condition	Post-Condition	Pre-Obligation	Post-Obligations
13a.ApproveProposalByDepartmentChair-Rule13a	Approve By Department Chair (Permit)	ApprovedByDepartmentChair = READYFORAPPROVAL position.title = Department Chair proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllChairs = false	all department chairs have not signed	Department Chair signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair
13b.ApproveProposalByDepartmentChair-Rule13b	Approve By Department Chair (Permit)	ApprovedByDepartmentChair = READYFORAPPROVAL position.title = Department Chair proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllChairs = true irbApprovalRequired = false	if all department chairs have signed then ApprovedByDepartmentChair = APPROVED ApprovedByBusinessManager = READYFORAPPROVAL	Department Chair signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Business Manager
13c.ApproveProposalByDepartmentChair-Rule13c	Approve By Department Chair (Permit)	ApprovedByDepartmentChair = READYFORAPPROVAL position.title = Department Chair	if all department	Department Chair signs the proposal	System sends an email to PI,

		proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllChairs = true irbApprovalRequired = true	chairs have signed then ApprovedBy DepartmentChair = APPROVED ApprovedBy BusinessManager = READYFOR APPROVAL (if IRB required then ApprovedBy IRB = READYFOR APPROVAL)		Co-PI, Senior Personnel, IRB and Business Manager
14a.ApproveProposalByBusinessManager-Rule14a	Approve By Business Manager (Permit)	ApprovedByBusinessManager = READYFORAPPROVAL position.title = Business Manager proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllBusinessManagers = false	All Business Managers have not signed.	Business Manager signs the proposal	System sends an email to PI, Co-PI and Senior Personnel, Business Manager

14b.ApproveProposalByBusinessManager-Rule14b	Approve By Business Manager (Permit)	ApprovedByBusinessManager = READYFORAPPROVAL position.title = Business Manager proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllBusinessManagers = true irbApprovalRequired = false	If all Business Managers have signed, then ApprovedByBusinessManager = APPROVED ApprovedByDean = READYFORAPPROVAL	Business Manager signs the proposal	System sends an email to PI, Co-PI and Senior Personnel, Dean
14c.ApproveProposalByBusinessManager-Rule14c	Approve By Business Manager (Permit)	ApprovedByBusinessManager = READYFORAPPROVAL position.title = Business Manager proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllBusinessManagers = true irbApprovalRequired = true ApprovedByIRB = READYFORAPPROVAL	If all Business Managers have signed, then ApprovedByBusinessManager = APPROVED ApprovedByDean = READYFORAPPROVAL	Business Manager signs the proposal	System sends an email to PI, Co-PI and Senior Personnel, Dean and IRB

			(if IRB required then ApprovedBy IRB = READYFOR APPROVAL)		
14d.ApproveProposalByBusinessManager-Rule14d	Approve By Business Manager (Permit)	ApprovedByBusinessManager = READYFORAPPROVAL position.title = Business Manager proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllBusinessManagers = true irbApprovalRequired = true ApprovedByIRB = APPROVED	If all Business Managers have signed, then ApprovedBy BusinessMan ager = APPROVED ApprovedBy Dean = READYFOR APPROVAL (if IRB required then ApprovedBy IRB = APPROVED)	Business Manager signs the proposal	System sends an email to PI, Co-PI and Senior Personnel, Dean

15a.ApproveProposalByDean-Rule15a	Approve By Dean (Permit)	ApprovedByDean = READYFORAPPROVAL position.title = Dean proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllDeans = false	All Deans have not signed	Dean signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Dean
15b.ApproveProposalByDean-Rule15b	Approve By Dean (Permit)	ApprovedByDean = READYFORAPPROVAL position.title = Dean proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllDeans = true irbApprovalRequired = false	If all Deans have signed, then ApprovedByDean = APPROVED ApprovedByUniversityResearchAdministrator = READYFORAPPROVAL	Dean signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, University Research Administrator
15c.ApproveProposalByDean-Rule15c	Approve By Dean (Permit)	ApprovedByDean = READYFORAPPROVAL position.title = Dean proposal.section = Whole Proposal proposal.action = Approve Condition:	If all Deans have signed, then ApprovedByDean = APPROVED If	Dean signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, IRB

		signedByAllDeans = true irbApprovalRequired = true ApprovedByIRB = READYFORAPPROVAL	ApprovedByIRB = READYFORAPPROVAL then ApprovedByIRB = APPROVED)		
15d.ApproveProposalByDean-Rule15d	Approve By Dean (Permit)	ApprovedByDean = READYFORAPPROVAL position.title = Dean proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllDeans = true irbApprovalRequired = true ApprovedByIRB = APPROVED	If all Deans have signed, then ApprovedByDean = APPROVED If ApprovedByIRB = APPROVED then ApprovedByUniversityResearchAdministrator = READYFORAPPROVAL)	Dean signs the proposal	System sends an email to PI, Co-PI, Senior Personnel and University Research Administrator

16a.ApproveProposalByIRB-Rule16a	Approve By IRB (Permit)	ApprovedByIRB = READYFORAPPROVAL position.title = IRB proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllIRBs = false irbApprovalRequired = true	All IRBs have not signed	IRB signs the proposal	System sends an email to PI, Co-PI, Senior Personnel and IRB
16b.ApproveProposalByIRB-Rule16b	Approve By IRB (Permit)	ApprovedByIRB = READYFORAPPROVAL position.title = IRB proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllIRBs = trueirbApprovalRequired = true approvedbydean = APPROVED	If all IRBs have signed, then ApprovedByIRB = APPROVED (if ApprovedByDean = APPROVED then ApprovedByUniversityResearchAdministrator = READYFOR	IRB signs the proposal	System sends an email to PI, Co-PI, Senior Personnel and University Research Administrator

			APPROVAL)		
16c.ApproveProposalByIRB-Rule16c	Approve By IRB (Permit)	ApprovedByIRB = READYFORAPPROVAL position.title = IRB proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllIRBs = true irbApprovalRequired = true ApprovedByBusinessManager = APPROVED	If all IRBs have signed, then ApprovedBy IRB = APPROVED (if ApprovedBy BusinessManager = APPROVED then ApprovedBy UniversityResearchAdministrator = READYFOR APPROVAL)	IRB signs the proposal	System sends an email to PI, Co-PI, Senior Personnel and University Research Administrator
16d.ApproveProposalByIRB-Rule16d	Approve By IRB (Permit)	ApprovedByIRB = READYFORAPPROVAL position.title = IRB proposal.section = Whole Proposal proposal.action = Approve	If all IRBs have signed, then ApprovedBy IRB =	IRB signs the proposal	System sends an email to PI, Co-PI, Senior Personnel,

		Condition: signedByAllIRBs = true irbApprovalRequired = true ApprovedByBusinessManager = READYFORAPPROVAL	APPROVED (if ApprovedBy BusinessMan ager = READYFOR APPROVAL then ApprovedBy BusinessMan ager = APPROVED)		Business Manager
16e.ApproveProposalByI RB-Rule16e	Approve By IRB (Permit)	ApprovedByIRB = READYFORAPPROVAL position.title = IRB proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllIRBs = true irbApprovalRequired = true approvedbydean = READYFORAPPROVAL	If all IRBs have signed, then ApprovedBy IRB = APPROVED (if ApprovedBy Dean = READYFOR APPROVAL then	IRB signs the proposal	System sends an email to PI, Co-PI, Senior Personnel and Dean

			ApprovedBy Dean = APPROVED)		
17a1.ApproveProposalBy UniversityResearchAdministrator-Rule17a1	Approve By University Research Administrator (Permit)	ApprovedByUniversityResearchAdministrator = READYFORAPPROVAL position.title = University Research Administrator proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllResearchAdmins = false	All University Research Administrators have not signed	University Research Administrator signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, University Research Administrator
17a2.ApproveProposalBy UniversityResearchAdministrator-Rule17a2	Approve By University Research Administrator (Permit)	ApprovedByUniversityResearchAdministrator = READYFORAPPROVAL position.title = University Research Administrator proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllResearchAdmins = true	if all University Research Administrators have signed, then ApprovedBy UniversityResearchAdministrator = APPROVED ApprovedBy UniversityResearchDirect	University Research Administrator signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, University Research Director

			or = READYFOR APPROVAL		
18a1.ApproveProposalBy UniversityResearchDirect or-Rule18a1	Approve by University Research Director (Permit)	ApprovedByUniversityResearchDirector = READYFORAPPROVAL position.title = University Research Director proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllResearchDirectors = false	All University Research Directors have not signed	University Research Director signs the proposal	System sends an email to PI, Co-PI, Senior Personnel and University Research Director
18a2.ApproveProposalBy UniversityResearchDirect or-Rule18a2	Approve by University Research Director (Permit)	ApprovedByUniversityResearchDirector = READYFORAPPROVAL position.title = University Research Director proposal.section = Whole Proposal proposal.action = Approve Condition: signedByAllResearchDirectors = true	If all University Research Directors have signed, then ApprovedBy UniversityRe searchDirect or = APPROVED	University Research Director signs the proposal	System sends an email to PI, Co-PI, Senior Personnel and University Research Administrator

Action: Disapprove					
Rule	**Action**	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligations**

19. DisapproveProposalByDepartmentChair-Rule19	Disapprove by Department Chair (Permit)	ApprovedByDepartmentChair = READYFORAPPROVAL position.title = Department Chair proposal.section = Whole Proposal proposal.action = Disapprove	ApprovedByDepartmentChair = DISAPPROVED SubmittedByPI = NOTSUBMITTED If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature	Department Chair signs the proposal	System sends email to PI, Co-PI and Senior Personnel, Department Chair
20a1.DisapproveProposalByBusinessManager-Rule20a1	Disapprove By Business Manager (Permit)	ApprovedByBusinessManager = READYFORAPPROVAL position.title = Business Manager proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = false	ApprovedByBusinessManager = DISAPPROVED SubmittedByPI = NOTSUBMITTED	Business Manager signs the proposal	System sends email to PI, Co-PI and Senior Personnel, Department Chair and Business Manager

			ApprovedBy IRB = NOTREADYFORAPPROVAL If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature		
20a2.DisapproveProposal ByBusinessManager-Rule20a2	Disapprove By Business Manager (Permit)	ApprovedByBusinessManager = READYFORAPPROVAL position.title = Business Manager proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = true ApprovedByIRB = APPROVED	ApprovedBy BusinessManager = DISAPPROVED SubmittedBy PI = NOTSUBMITTED ApprovedBy IRB = NOTREADYFORAPPROVAL	Business Manager signs the proposal	System sends email to PI, Co-PI and Senior Personnel, Department Chair Business Manager, IRB

			If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature		
20a3.DisapproveProposalByBusinessManager-Rule20a2	Disapprove By Business Manager (Permit)	ApprovedByBusinessManager = READYFORAPPROVAL position.title = Business Manager proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = true ApprovedByIRB = READYFORAPPROVAL	ApprovedByBusinessManager = DISAPPROVED SubmittedByPI = NOTSUBMITTED ApprovedByIRB = NOTREADYFORAPPROVAL If Co-PI>0 ReadyForSubmissionByPI = False	Business Manager signs the proposal	System sends email to PI, Co-PI and Senior Personnel, Department Chair Business Manager, IRB

			Clear all signature		
21a1.DisapproveProposal ByDean-Rule21a	Disapprove by Dean (Permit)	ApprovedByDean = READYFORAPPROVAL position.title = Dean proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired =false	ApprovedBy Dean = DISAPPROVED SubmittedBy PI = NOTSUBMITTED If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature	Dean signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business Manager and Dean
21a2.DisapproveProposal ByDean-Rule21a2	Disapprove by Dean (Permit)	ApprovedByDean = READYFORAPPROVAL position.title = Dean proposal.section = Whole Proposal proposal.action = Disapprove Condition:	ApprovedBy Dean = DISAPPROVED SubmittedBy PI = NOTSUBMITTED	Dean signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business

		irbApprovalRequired = true ApprovedByIRB = APPROVED	If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature		Manager, IRB and Dean
21a3.DisapproveProposalByDean-Rule21a3	Disapprove by Dean (Permit)	ApprovedByDean = READYFORAPPROVAL position.title = Dean proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = true ApprovedByIRB = READYFORAPPROVAL	ApprovedByDean = DISAPPROVED SubmittedByPI = NOTSUBMITTED If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature	Dean signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, IRB and Dean
22a.DisapproveProposalByIRB-Rule22a	Disapprove By IRB (Permit)	ApprovedByIRB = READYFORAPPROVAL position.title = IRB	ApprovedByIRB =	IRB signs the proposal	System sends an email to PI, Co-PI,

		proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = true	DISAPPROVED SubmittedByPI = NOTSUBMITTED If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature		Senior Personnel, Department Chair and IRB
22b.DisapproveProposalByIRB-Rule22b	Disapprove By IRB (Permit)	ApprovedByIRB = READYFORAPPROVAL position.title = IRB proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = true ApprovedByBusinessManager = APPROVED	ApprovedByIRB = DISAPPROVED SubmittedByPI = NOTSUBMITTED If Co-PI>0 ReadyForSubmissionByPI = False	IRB signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair, IRB and Business Manager

			Clear all signature		
22c.DisapproveProposalByIRB-Rule22c	Disapprove By IRB (Permit)	ApprovedByIRB = READYFORAPPROVAL position.title = IRB proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = true ApprovedByBusinessManager = READYFORAPPROVAL	ApprovedByIRB = DISAPPROVED SubmittedByPI = NOTSUBMITTED If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature	IRB signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair, IRB and Business Manager
22d.DisapproveProposalByIRB-Rule22d	Disapprove By IRB (Permit)	ApprovedByIRB = READYFORAPPROVAL position.title = IRB proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = true	ApprovedByIRB = DISAPPROVED SubmittedByPI = NOTSUBMITTED	IRB signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business

		ApprovedByDean = APPROVED	If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature		Manager, Dean and IRB
22e.DisapproveProposalByIRB-Rule22e	Disapprove By IRB (Permit)	ApprovedByIRB = READYFORAPPROVAL position.title = IRB proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = true ApprovedByDean = READYFORAPPROVAL	ApprovedByIRB = DISAPPROVED SubmittedByPI = NOTSUBMITTED If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature	IRB signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean and IRB
23a1.DisapproveProposalByUniversityResearchAdministrator-Rule23a1	Disapprove By University Research	ApprovedByUniversityResearchAdministrator = READYFORAPPROVAL	ApprovedByUniversityRe	University Research Administrat	System sends an email to PI, Co-PI,

	Administrator (Permit)	position.title = University Research Administrator proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = false	searchAdministrator = DISAPPROVED SubmittedByPI = NOTSUBMITTED If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature	or signs the proposal	Senior Personnel, Department Chair, Business Manager,Dean, University Research Administrator
23a2.DisapproveProposalByUniversityResearchAdministrator-Rule23a2	Disapprove By University Research Administrator (Permit)	ApprovedByUniversityResearchAdministrator = READYFORAPPROVAL position.title = University Research Administrator proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = true	ApprovedByUniversityResearchAdministrator = DISAPPROVED SubmittedByPI = NOTSUBMITTED	University Research Administrator signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University

			If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature		Research Administrator and IRB
24a1.DisapproveProposalByUniversityResearchDirector-Rule24a1	Disapprove by University Research Director (Permit)	ApprovedByUniversityResearchDirector = READYFORAPPROVAL position.title = University Research Director proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = false	ApprovedByUniversityResearchDirector = DISAPPROVED SubmittedByPI = NOTSUBMITTED If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature	University Research Director signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Administrator, University Research Director

24a2.DisapproveProposalByUniversityResearchDirector-Rule24a2	Disapprove by University Research Director (Permit)	ApprovedByUniversityResearchDirector = READYFORAPPROVAL position.title = University Research Director proposal.section = Whole Proposal proposal.action = Disapprove Condition: irbApprovalRequired = false	ApprovedByUniversityResearchDirector = DISAPPROVED SubmittedByPI = NOTSUBMITTED If Co-PI>0 ReadyForSubmissionByPI = False Clear all signature	University Research Director signs the proposal	System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB

Action: Withdraw					
Rule	**Action**	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligations**
25a1.WithdrawProposalByUniversityResearchAdministrator-Rule25a1	Withdraw By University Research	WithdrawnByUniversityResearchAdministrator = NOTWITHDRAWN	WithdrawnByUniversityResearchAdmi		System sends an email to PI, Co-PI,

	Administrator (Permit)	ApprovedByUniversityResearchAdministrator = READYFORAPPROVAL position.title = University Research Administrator proposal.section = Whole Proposal proposal.action = Withdraw Condition: irbApprovalRequired = false	nistrator = WITHDRAWN ApprovedByUniversityResearchAdministrator = NOTREADYFORAPPROVAL		Senior Personnel Department Chair, Business Manager, Dean, University Research Administrator, University Research Director
25a2.WithdrawProposalByUniversityResearchAdministrator-Rule25a2	Withdraw By University Research Administrator (Permit)	WithdrawnByUniversityResearchAdministrator = NOTWITHDRAWN ApprovedByUniversityResearchAdministrator = READYFORAPPROVAL position.title = University Research Administrator proposal.section = Whole Proposal proposal.action = Withdraw Condition: irbApprovalRequired = true	WithdrawnByUniversityResearchAdministrator = WITHDRAWN ApprovedByUniversityResearchAdministrator = NOTREADYFORAPPROVAL		System sends an email to PI, Co-PI, Senior Personnel Department Chair, Business Manager, Dean, University Research Administrator, University

					Research Director and IRB

Action: Archive					
Rule	**Action**	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligations**
26a1.ArchiveProposalByUniversityResearchDirector-Rule26a1	Archive By University Research Director (Permit)	ArchivedByUniversityResearchDirector = NOTARCHIVED SubmittedByUniversityResearchAdministrator = SUBMITTED position.title = University Research Director proposal.section = Whole Proposal proposal.action = Archive Condition: irbApprovalRequired = false	ArchivedByUniversityResearchDirector = ARCHIVED ApprovedByUniversityResearchDirector = NOTREADYFORAPPROVAL		System sends an email to PI, Co-PI, Senior Personnel Department Chair, Business Manager, Dean, University Research Administrator and University Director
26a2.ArchiveProposalByUniversityResearchDirector-Rule26a2	Archive By University Research Director (Permit)	ArchivedByUniversityResearchDirector = NOTARCHIVED SubmittedByUniversityResearchAdministrator = SUBMITTED position.title = University Research Director	ArchivedByUniversityResearchDirector = ARCHIVED ApprovedByUniversityResearchDirector =		System sends an email to PI, Co-PI, Senior Personnel Department Chair, Business

		proposal.section = Whole Proposal proposal.action = Archive	NOTREADYFORAPPROVAL		Manager, Dean, University Research Administrator University Research Director and IRB

Action: Delete					
Rule	**Action**	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligations**
27. DeleteCo-PIandSeniorPersonnelByPI-Rule27	Co-PI can be deleted by PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED proposal.section = InvestigatorInformation.Co-PI proposal.action = Delete			
	Senior Personnel can be Deleted by PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED proposal.section = InvestigatorInformation.Senior-Personnel proposal.action = Delete			
28. DeleteSeniorPersonnelByCoPI-Rule28	Senior Personnel can be Deleted by Co-PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED ReadyForSubmissionByPI = False			

		proposal.section = InvestigatorInformation.Senior-Personnel proposal.action = Delete			
29. CannotDeleteCoPIByCoPI-Rule29	Co-PI cannot be Deleted by Co-PI (Deny)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED ReadyForSubmissionByPI = False proposal.section = InvestigatorInformation.Co-PI proposal.action = Delete			
30. DeleteProposalByPI-Rule30	Delete Proposal by PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED proposal.role = PI proposal.section = Whole Proposal proposal.action = Delete			System sends email to PI, Co-PI, Senior Personnel
31. CannotDeleteProposalByCo-PI-Rule31	Not delete Proposal by Co-PI (Deny)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED proposal.role = Co-PI proposal.section = Whole Proposal proposal.action = Delete			
32a1.DeleteProposalByUniversityResearchDirector-Rule32a1	Delete by University Research Director (Permit)	DeletedByUniversityResearchDirector = NOTDELETED ApprovedByUniversityResearchDirector = READYFORAPPROVAL position.title = University Research Director proposal.section = Whole Proposal proposal.action = Delete	DeletedByUniversityResearchDirector = DELETED ApprovedByUniversityResearchDirector =		System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business

		Condition: irbApprovalRequired = false	NOTREADYFORAPPROVAL		Manager, Dean, University Research Administrator, University Research Director
32a2.DeleteProposalByUniversityResearchDirector-Rule32a2	Delete by University Research Director (Permit)	DeletedByUniversityResearchDirector = NOTDELETED ApprovedByUniversityResearchDirector = READYFORAPPROVAL position.title = University Research Director proposal.section = Whole Proposal proposal.action = Delete Condition: irbApprovalRequired = true	DeletedByUniversityResearchDirector = DELETED ApprovedByUniversityResearchDirector = NOTREADYFORAPPROVAL		System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB

Action: View					
Rule	**Action**	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligations**
33. ViewAuditLogBy PI-Rule33	Audit Log View by PI (Permit)	proposal.section = Audit Log proposal.role = PI proposal.action = View			
34. CannotViewAudit LogByOtherUser-Rule34	AuditLog not view by Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, IRB, University Research Administrator, University Research Director (Deny)	proposal.section = Audit Log proposal.role = Co-PI || Senior Personnel || position.title = Department Chair || Business Manager || Dean || IRB || University Research Administrator || University Research Director proposal.action = View			

Action: Edit					
Rule	**Action**	**Pre-Condition**	**Post-Condition**	**Pre-Obligation**	**Post-Obligations**
35. EditProposalSectio nByPI-Rule35a	Proposal Section Edit by PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED proposal.section = <Whole Proposal || Investigator Information || Project			

		Information || Sponsor and Budget Information || Cost Share Information || University Commitments || Conflict of Interest and Commitment Information || Compliance Information || Additional Information || Collaboration Information || Proprietary/Confidential Information || Certification/Signatures || Appendices> proposal.role = PI proposal.action = Edit			
36. CannotEditOSPSectionByPI-Rule36	PI Cannot Edit OSP section (Deny)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED proposal.section = OSP section proposal.role = PI proposal.action = Edit			
37. EditProposalSectionByCoPI-Rule37	Proposal Section Edit by Co-PI (Permit)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED ReadyForSubmissionByPI = False proposal.section = <Investigator Information || Certification/Signatures || Appendices > proposal.action = Edit			
38. CannotEditSomeProposalSectionByCoPI-Rule38	Co-PI cannot Edit Project Information, Sponsor and Budget Information, Cost Share Information,	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED ReadyForSubmissionByPI = False proposal.section = <Project Information || Sponsor and Budget Information || Cost			

	University Commitments, Conflict of Interest and Commitment Information, Compliance Information, Additional Information, Collaboration Information, Proprietary/Confidential Information, OSP Section (Deny)	Share Information || University Commitments || Conflict of Interest and Commitment Information || Compliance Information || Additional Information || Collaboration Information || Proprietary/Confidential Information || OSP Section> proposal.action = Edit			
39. CannotEditProposalSectionBySeniorPersonnel-Rule39	Proposal section not Edit by Senior Personnel (Deny)	SubmittedByPI = NOTSUBMITTED DeletedByPI = NOTDELETED ReadyForSubmissionByPI = False proposal.section = <Investigator Information || Project Information || Sponsor and Budget Information || Cost Share Information || University Commitments || Conflict of Interest and Commitment Information || Compliance Information || Additional Information || Collaboration Information || Proprietary/Confidential Information || Certification/Signatures || OSP Section || Appendices>			

		proposal.role = Senior Personnel proposal.action = Edit			
40. EditProposalSectionByDepartmentChair-Rule40	Certification/Signatures edit by Department Chair (Permit)	ApprovedByDepartmentChair = READYFORAPPROVAL proposal.section = Certification/Signatures position.title = Department Chair proposal.action = Edit			
41. CannotEditProposalSectionByDepartmentChair-Rule41	Proposal Section not edit by Department Chair (Deny)	ApprovedByDepartmentChair = READYFORAPPROVAL proposal.section = <Investigator Information || Project Information || Sponsor and Budget Information || Cost Share Information || University Commitments || Conflict of Interest and Commitment Information || Compliance Information || Additional Information || Collaboration Information || Proprietary/Confidential Information || OSP Section || Appendices> position.title = Department Chair proposal.action = Edit			
42. EditProposalSectionByBusinessManager-Rule42	Edit by Business Manager (Permit)	ApprovedByBusinessManager = READYFORAPPROVAL proposal.section = <Sponsor and Budget Information || Certification/Signatures> position.title = Business Manager proposal.action = Edit			

43. CannotEditProposalSectionByBusinessManager-Rule43	Not edit by Business Manager (Deny)	ApprovedByBusinessManager = READYFORAPPROVAL proposal.section = <Investigator Information, Project Information, Cost Share Information, University Commitments, Conflict of Interest and Commitment Information, Compliance Information, Additional Information, Collaboration Information, Proprietary/Confidential Information, OSP Section, Appendices> position.title = Business Manager proposal.action = Edit			
44. EditProposalSectionByDean-Rule44	Certification/Signatures edit by Dean (Permit)	ApprovedByDean = READYFORAPPROVAL proposal.section = Certification/Signatures position.title = Dean proposal.action = Edit			
45. CannotEditProposalSectionByDean-Rule45	Proposal Section not edit by Dean (Deny)	ApprovedByDean = READYFORAPPROVAL proposal.section = <Investigator Information || Project Information || Sponsor and Budget Information || Cost Share Information || University Commitments || Conflict of Interest and Commitment Information || Compliance Information || Additional Information || Collaboration			

		Information || Proprietary/Confidential Information || OSP Section || Appendices> position.title = Dean proposal.action = Edit			
46. EditProposalSectionByIRB-Rule46	Certification/Signatures edit by IRB (Permit)	ApprovedByIRB = READYFORAPPROVAL proposal.section = Certification/Signatures position.title = IRB proposal.action = Edit			
47. CannotEditProposalSectionByIRB-Rule47	Proposal Section not edit by IRB (Deny)	ApprovedByIRB = READYFORAPPROVAL proposal.section = <Investigator Information || Project Information || Sponsor and Budget Information || Cost Share Information || University Commitments || Conflict of Interest and Commitment Information || Compliance Information || Additional Information || Collaboration Information || Proprietary/Confidential Information || OSP Section || Appendices> position.title = IRB proposal.action = Edit			
48. EditProposalSectionByUniversityResearchAdministrator-Rule48	Proposal Section edit by University Research Administrator (Permit)	ApprovedByUniversityResearchAdministrator = READYFORAPPROVAL Proposal.section = <Investigator Information || Project Information || Sponsor and Budget Information || Cost Share			

		Information || University Commitments || Conflict of Interest and Commitment Information || Compliance Information || Additional Information || Collaboration Information || Proprietary/Confidential Information || OSP Section || Certification/Signatures> position.title = University Research Administrator proposal.action = Edit			
49. CannotEditProposalSectionByUniversityResearchAdministrator-Rule49	Appendices not edit by University Research Administrator (Deny)	ApprovedByUniversityResearchAdministrator = READYFORAPPROVAL Proposal.section = Appendices position.title = University Research Administrator proposal.action = Edit			
50. EditProposalSectionByUniversityResearchDirector-Rule50	Proposal Section edit by University Research Director (Permit)	ApprovedByUniversityResearchDirector = READYFORAPPROVAL Proposal.section = Certification/Signatures || OSP Section position.title = University Research Director proposal.action = Edit			
51. CannotEditProposalSectionByUniversityResearchDirector-Rule51	Proposal Section not edit by University Research Director (Deny)	ApprovedByUniversityResearchDirector = READYFORAPPROVAL Proposal.section = <Investigator Information || Project Information || Sponsor and Budget Information || Cost Share			

		Information, University Commitments || Conflict of Interest and Commitment Information, Compliance Information || Additional Information || Collaboration Information || Proprietary/Confidential Information || Appendices> position.title = University Research Director proposal.action = Edit			

APPENDIX E

Policy Requirement Description

1. A “Tenured/Tenured-track faculty” is allowed to add a new “Proposal”.

2. “PI” can “Delete” a “Whole Proposal” when SubmittedByPI = NOTSUBMITTED and not been already deleted without any pre-obligation but with Post-obligation: Send Email to all Investigators such as PI, CO-PIs, and Senior Personnel.

3. “Department Chair” can “Approve” a “Whole Proposal” when ApprovedByDepartmentChair = READYFORAPPROVAL with Pre-obligation: Chair needs to Sign it first and Post-obligation: Send Email to all Investigators such as PI, CO-PIs, and Senior Personnel.

4. “Department Chair” can “Delegate” his actions “Approve/Disapprove” to “Associate Chair” from his own Department when ApprovedByDepartmentChair = READYFORAPPROVAL.

5. “Associate Chair” can “Approve” proposal when ApprovedByDepartmentChair = READYFORAPPROVAL with Conditions: Delegation is active with Pre-obligation: Chair needs to Sign it first and Post-obligation: Send Email to all Investigators such as PI, CO-PIs, and Senior Personnel.

APPENDIX F

Policy Rule with Obligation

```
<?xml version="1.0" encoding="UTF-8"?>
- <PolicySet xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides" Version="1.0" PolicySetId="PolicySet1">
    <Description>PolicySet for GPMS.</Description>
    <Target/>
  - <Policy Version="1.0" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides" PolicyId="Proposal-Rules">
        <Description>Policy for any proposal.</Description>
      + <PolicyDefaults>
        <Target/>
      - <Rule RuleId="ApproveProposalByDepartmentChair-Rule13b" Effect="Permit">
            <Description>"Department Chair" can "Approve" a "Whole Proposal" when ApprovedByDepartmentChair = READYFORAPPROVAL and where condition
                check all department chairs are approved and no IRB is required.</Description>
          - <Target>
            - <AnyOf>
              - <AllOf>
                + <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                + <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                + <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                + <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                </AllOf>
              </AnyOf>
            </Target>
          - <Condition>
            - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
                + <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-equal">
                + <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-equal">
              </Apply>
            </Condition>
          - <ObligationExpressions>
              + <ObligationExpression FulfillOn="Permit" ObligationId="sendAlert">
              + <ObligationExpression FulfillOn="Permit" ObligationId="sendEmail">
            </ObligationExpressions>
        </Rule>
    </Policy>
</PolicySet>
```

APPENDIX G

XACML Request Format example

```
<?xml version="1.0" encoding="UTF-8"?>
- <Request ReturnPolicyIdList="false" CombinedDecision="false" xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
  - <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
    - <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:position.title">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Department Chair</AttributeValue>
      </Attribute>
    </Attributes>
  - <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
    - <Content>
      - <ak:record xmlns:ak="http://akpower.org">
          + <ak:proposal>
        </ak:record>
      </Content>
    - <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector">
        <AttributeValue DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-
          category:resource">//ak:record/ak:proposal</AttributeValue>
      </Attribute>
    - <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:resource:ApprovedByDepartmentChair">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">READYFORAPPROVAL</AttributeValue>
      </Attribute>
    - <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:resource:proposal.section">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Whole Proposal</AttributeValue>
      </Attribute>
    </Attributes>
  - <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
    - <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:action:proposal.action">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Approve</AttributeValue>
      </Attribute>
    </Attributes>
</Request>
```

APPENDIX H

XACML Response Format example with Obligations

```
<?xml version="1.0"?>
- <Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
  - <Result>
      <Decision>Permit</Decision>
    - <Status>
        <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
      </Status>
    - <Obligations>
      - <Obligation ObligationId="sendAlert">
          <AttributeAssignment DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="obligationType">preobligation</AttributeAssignment>
          <AttributeAssignment DataType="http://www.w3.org/2001/XMLSchema#boolean" AttributeId="signedByCurrentUser">true</AttributeAssignment>
          <AttributeAssignment DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="alertMessage">You need to sign the proposal first!
            </AttributeAssignment>
        </Obligation>
      - <Obligation ObligationId="sendEmail">
          <AttributeAssignment DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="obligationType">postobligation</AttributeAssignment>
        + <AttributeAssignment DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="emailBody">
          <AttributeAssignment DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="emailSubject">Your proposal has been approved by:
            </AttributeAssignment>
          <AttributeAssignment DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="authorName">Computer Science Department
            Chair</AttributeAssignment>
          <AttributeAssignment DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="piEmail">milsonmun@gmail.com</AttributeAssignment>
          <AttributeAssignment DataType="http://www.w3.org/2001/XMLSchema#string"
            AttributeId="managersEmail">bmcomputerscience@gmail.com</AttributeAssignment>
        </Obligation>
      </Obligations>
  </Result>
</Response>
```