Major Project Report on

Design and Development of a Workflow Management System Using UML And C++

Submitted in Partial fulfillment of
The requirement for the award of Degree of

MASTER OF ENGINEERING
in
COMPUTER TECHNOLOGY AND APPLICATIONS

by

SIMMI DUTTA
25/CTA/O3
M.E. (Computer Technology and Applications)
Delhi College of Engineering

Under the guidance of

Dr. GOLDIE GABRANI
Delhi College of Engineering

DEPARTMENT OF COMPUTER ENGINEERING
DELHI COLLEGE OF ENGINEERING
UNIVERSITY OF DELHI, DELHI-110042

CERTIFICATE

This is to certify that the Thesis entitled "**DESIGN AND DEVELOPMENT OF A WORKFLOW MANAGEMENT SYSTEM USING UML AND C++"** submitted by Simmi Dutta towards the partial fulfillment of requirement for the degree of Master of Engineering in Computer Technology and Applications is a bonafide record of her work carried out under the supervision and guidance of Dr. Goldie Gabrani.

Further it is also certified that this project has not been submitted for any other degree or diploma in any other college to the best of our knowledge.

Dr. Goldie Gabrani
Asstt. Professor
Department of Computer Engg.
Delhi College of Engg.

Dr. D. Roy Choudhury
Professor & Head
Department of Computer Engg.
Delhi College of Engg.

Acknowledgement

I would first of all like to thank my thesis guide, **Dr. Goldie Gabrani**, Asstt. Professor, Deptt. Of Computer Engg., DCE, for her expert guidance, valuable advice and continuous involvement and encouragement during all the stages of the work. I would also like to express my gratitude to **Dr. D. Roy Choudhary**, Professor and head, Deptt. Of Computer Engg., DCE, for his constructive criticism and valuable guidance that have contributed a lot to the completion of my thesis.

I also owe a word of Sincere thanks to my **Parents,** husband **Sanjay** for their unending support and most specially my son **Anav**, who has sacrificed his precious two childhood years in the process of my higher studies. A word of sincere thanks is also due to my sister **Anita** and her family for providing the much needed homely environment in an alien city and also my brother **Rajeev** without whose efficient guidance, I would be just groping in the dark.

Last but not the least I would like to thank all my friends and batch mates, who directly or indirectly helped me in the completion of this thesis work.

SIMMI DUTTA

Abstract

In modern world all computational and information technology is striving to become more human centric and ergonomic. The complete process of information management is more than a pure transfer of information, it involves interaction with human beings where the latter may modify, modulate or control its flow. This is the concept behind a workflow management system. The information is essentially a workitem, which is acted upon by participants in the workflow. These participants may be humans or automated computer procedures. The WorkFlow Management System (WFMS) also provides process definition framework that models the flow of work in an organization and its instantiation. In the current work, the aim is to develop and implement a workflow management system. WFMS are a relatively new concept and it is intended to use open source software to implement a typical infrastructure. This thesis analyses the workflow in a typical engineering college and develops the models required in UML. These developed models in UML and the activity diagrams provide a starting point as a workflow definition in WFMS. This work used the computing infrastructure on a host system running open source Linux (Fedora Core 3) and java based open source workflow engine OpenWFE. The proposed framework is extensible and provides facility to implement more rigorous workflow solutions based on the departmental / user needs.

List of Acronyms

I T	Information Technology
BPR	Business Process Re-engineering
BPM	Business Process management
WFMS	Workflow Management System
WFMC	Workflow Management Coalition
API	Application Programming Interface
Open WFE	Open Workflow Engine
Apre	Automatic Participant runtime environment
JDK	Java Development Kit
JRE	Java Runtime Environment
JSP	Java Server Pages
JMS	Java Message Service
JVM	Java Virtual Machine
EJB	Enterprise Java Bean
SOA	Service Oriented Architecture
EAI	Enterprise Application Integration
BPEL4WS	Business Process Execution Language for Web Services
SQL	Structured Query Language
XML	Extensible Markup Language
XPDL	XML Process Definition Language
OMG	Object Management Group
OW4J	Oracle Workflow for Java
jPDL	jBPM Process Definition Language

List of Figures

List of Tables

Table of Contents

Chapter 1

Introduction

1.1 Basics of Workflow

What is Workflow?

In the middle Ages, monks sat at tables carefully copying the scriptures. The father superior would make the assignments, perhaps giving the illuminated first page of a section to the most skilled artist, perhaps assigning the proofreading tasks to the elderly scholar with trembling hands.

Little has changed in centuries—supervisors assign work, perhaps based on training, skills, and experience, to various resources. At first the resources were only people, eventually supported by tools such as typewriters, printed forms, and adding machines. Eventually some of the steps were automated—the invoices were automatically totaled and printed, but only after people sorted the punched cards, or entered the data. Even though the performance of the work was at least partially automated, the management of the work had changed little—supervisors assigned work and monitored performance. Clerks passed the work from station to station. Lists were made to

track the work—to find it when it went astray, and to measure the productivity. And an army of expediters searched for the problems and errors in the routing, and kept it moving.

In the last 15 years or so we finally have developed tools to not only do the work, but to manage the workflow. More than just procedural documents, that workflow process is defined formally in the workflow computer system. The process is managed by a computer program that assigns the work, passes it on, and tracks its progress.

The workflow process is traditionally defined in office terms—moving the paper, processing the order, issuing the invoice. But the same principles and tools apply to filling the order from the warehouse, assembling documents, parts, tools, and people to repair a complex system, manufacturing the complex device or registering for an academic course in a university or college.

With the automated workflow management system the following benefits accrue as given by Charles Plesums [1]:

(a) Work doesn't get misplaced or stalled—expediters are rarely required to recover from errors or mismanagement of the work·

(b) The managers can focus on staff and business issues, such as individual performance, optimal procedures, and specialists, rather than the routine assignment of tasks. The army of clerks is no longer required to deliver and track the work.

(c) The procedures are formally documented and followed exactly, ensuring that the work is performed in the way planned by management, meeting all business and regulatory requirements.

(d) The best person (or machine) is assigned to do each case, and the most important cases are assigned first. Users don't waste time choosing which item to work on, perhaps procrastinating on important but difficult cases.

(e) Parallel processing, where two or more tasks are performed concurrently, is far more practical than in a traditional, manual workflow.

With the best person doing the most important work following the correct procedures, not only is the business conducted more effectively, but also costs are lowered and the service to the customers is generally better. With the work equitably distributed and the confidence that they are always working on the “right” thing, users are happier. Therefore workflow is good for the company, good for the customers, and good for the users.

Why workflow?

Work done by the best participant

A simple workflow system could evenly distribute work among all the available resources, or follow a simple algorithm such as giving the waiting work to the resource with the shortest queue, or implement assignments made manually by a supervisor. However, there are often significant benefits when the system can optimize the assignment of the work. In order to do the assignment, the workflow management system must know who or what is available to perform the work, and have a profile about each user. This might include what work the resources are qualified to do, how good they are at that type of work (can they do only routine processing or can they handle the toughest cases), and whether the supervisor wants the work to be assigned to them. Generally a priority is calculated, based on the type of work and how long it has been waiting to be processed.

“Assembly Line” or “Once and Done”

Much work, even office procedures, can be processed in an assembly line, where there each step in the process is simple and specialized. Most of the steps are simple. Training is minimized. Staffing is often flexible, because few steps require specialized skills, authority, or licensing. However, an effort is required to move the work between steps, time is lost waiting at each step, and there are more chances that the work will be lost or misplaced. The other approach, equally popular among the experts, is to have highly trained staff handle the complete process, once and done. Far more training is required, with corresponding delegation of authority, but there is far less overhead in the work process itself. When the work is managed manually, once and done is generally superior, because there is less overhead and chance for error. But with an automated work management system, either approach can be used. The work management system tracks the

work, dramatically reducing the overhead of the assembly line process, with one remaining disadvantage. The elapsed time to complete the work (as seen by the customer) may be longer with the assembly line if a queue is allowed to build at each step in the process. Therefore in practice, a blend seems to be best—separate people to enter data and proofread, to reduce the chance of errors, with total automation of any step where that is practical. But minimize the number of steps where practical, to reduce the time waiting in queue, and thus reduce the total time of service. Once there is a system that tracks the multiple steps, perhaps not all of the steps need to be done sequentially. Manual systems often include a checklist or routing slip that accompanies the work, thus sequential processing is the easiest way, by far. The limitations of the checklist go away with an automated workflow system. Thus most systems allow multiple tasks to be assigned and performed concurrently, with a function to determine when all the parallel paths are complete, so the consolidated part of the workflow can continue.

Rendezvous

One of the tougher steps in the paper-processing world is waiting for a supporting document to arrive. In larger organizations the problem is harder since the recipient cannot remember every case, or there may be multiple people who could make the final decision. The documents need to be filed, and each arriving document that could satisfy a requirement needs to be checked that something isn't waiting for it. A list must be maintained—a tickler system—to trigger a follow up, if the missing information doesn't arrive in a timely manner. Most automated work management systems support the automatic matching of incoming information to the work that is suspended, waiting for the arrival of that information. The systems also manage the follow-up processing when the information does not arrive on time. The most common name is probably "rendezvous" but some systems call the process "marriage" or other terms.

Distribution

There are a variety of methods to distribute the work to the participants. Work may be pre-assigned, and then selected from the "in-box" by the user. The user may look through a common queue and "pull" the desired work. The user may ask the system to select and assign the most appropriate piece of work—to "push" an item of work. Or the work may be assigned based on "time." The ideal way to distribute work is to let the work management system assign the work,

following the rules to optimize that distribution. Sometimes called “Send Work”, “Get Work,” or “Assign Work,” in each case the system pushes work to the participant.

Vertical Workflow

One function of a workflow system is to get the work to the right person or process—to move the work through the organization. This is sometimes called “horizontal” workflow. Normally a case consists of multiple tasks—programs, portions of programs, or manual steps—performed at each step in the workflow. Therefore another possible workflow function is to see that all the tasks are completed—the “vertical” workflow.

Completion

When the work is completed, the normal conclusion is to change the status (from waiting for decision to either approved or rejected or from waiting for input to ready for checking). Each workflow product has its unique way to move the work to the next step in the process – not all use the term “status.”

Information easily accessible

As work is processed, computer systems and other data are often accessed. It is critical that the information necessary to complete the processing is easily accessible. With electronic assignment possible across multiple locations, the data may even be in a different city. Therefore it is common for a work management system to:

(a) Interface to existing computer databases and systems

(b) Invoke complex computer systems, possibly through terminal emulation.

(c) Link to document images, fax servers, e-mail, or other “external” sets of data.

(d) Extract key information to move with the workflow—for example, the items in an order, a credit limit, the limits of an insurance policy, or the current shipping address.

Interface to the data systems

Many organizations are concerned with the complexity of interfacing the workflow system to the business application, or even feel the need to integrate workflow with the application. On the contrary, there are many levels of potential interface. Workflow can be used with no interface to

the legacy processing systems. In practice, a minimal interface, to invoke particular programs and enter key data is popular with users, inexpensive to implement, and contributes to productivity. If the users only process a single application, it is possible that the embedded workflow approach, where the application drives the workflow, is sufficient. In practice, most users today must support multiple applications—perhaps an order entry system. It may be more practical to have a workflow system that invokes the appropriate legacy computer system, rather than having one legacy system invoke workflow.

Image systems

Work management systems are often installed in conjunction with document image systems. If there are paper documents involved in the workflow, it doesn't do much good to make an assignment and then have to search for the paper. On the other hand, if the necessary information is available in another form, there is no requirement for an image system with every work management system. Image systems often are installed in conjunction with work management systems. Therefore we should conclude that although image and work management systems often go together, and often bring synergistic benefits to the other; neither requires the other in all cases.

Logging and tracking

Workflow systems typically record the processing history, and provide the opportunity for the users to enter comments. The history typically includes the date, time, person where each step was performed, including the disposition of the step—for example, was the process approved and moved on in the workflow, or was the work suspended for later processing.

Search for work in process

Messages are often received—"What is the status?" or "Change my Request" or "Cancel the order." A work management system must not only manage the work in process, but must also identify the work so that it can be found. Not only must it know that this is a queue of 200 orders, but it must be able to find a specific order in that queue. And once the order has been found, we must be able to determine the status of that work.

Control

One of the big advantages of an automated work management system is the control of the process, manifest by the procedures that are implemented by the system, and the record keeping to report on the process. In all cases, the user profiles—qualifications, assignments, absences, vacations, training, and other factors are maintained by the system. Managers need to be able to manually assign work to an individual—perhaps because they made an error and need to fix it, or perhaps because it is a special case that they are uniquely qualified to fix. Priorities must be adjusted to move critical work to the head of the list.

Monitoring

Practically all systems include reporting and analysis such as the total work accomplished—the volume, and the turn-around time. The systems also maintain data to report the productivity of the individuals, teams, and groups. A few systems even maintain data about the number and types of errors that are caught and corrected for each type of process and user. Work management systems allow managers to examine the backlog of work throughout the day, in real time, so that they can schedule staff as required, adjust assignments if necessary to meet deadlines, and in general, manage their teams.

1.2 Measuring the Benefits

The benefits of a work management system have been divided into three categories by Charles Plesums [1]. The direct cost savings are readily measured and recognized. But there is another set of benefits that are real and valuable but very difficult to measure, sometimes called the hidden savings. And still more benefits where the value cannot be quantified—the intangible benefits.

Direct Cost Savings

These are the readily measured benefits. Often they involve better use of staff, or reduction of the staff.

Hidden Savings

The hidden savings are actual cash savings, but those that are far harder to measure. For example, better control of the work, savings of manager time, improved productivity of the professional staff, and the opportunity for process improvement.

Intangible Benefits

There are a number of benefits of work management that most people cannot quantify. Unlike the hidden benefits that are real and, with enough effort, can be quantified, it is rare that a cash value can be assigned to improved service, employee satisfaction, organizational options, security, and privacy. If any of the areas have a quantified value in your company, great—that factor can simply be added to the direct or hidden savings.

1.3 Workflow Types

For many years, business analysts and authors categorized workflow systems. Although such categories have fallen into disfavor, they are still instructive—they may help understand the differences between various systems. The various categories as given by Charles Plesums and Rob Allen [1,2] are discussed below.

Adhoc

Adhoc, or collaborative, workflow is often used in the professional and administrative areas of an organization. It is characterized by negotiation, and a new workflow defined for each use. This type of workflow is tremendously convenient, and provides good control of the process .The ad-hoc workflows are often built on an e-mail platform.

Production

A production workflow is predefined and prioritized, and thus supports high volumes—there are no negotiations about who will do the work or how it will be handled. However, there may be additional tasks or workflows defined and added to the overall process. The production workflow

can be very simple or complex. They can be completely predefined, or follow a general procedure, with additional steps and processes added as required. They can be altered for consultations. Normally it has a dedicated delivery channel, rather than using e-mail to deliver the work. Overall the production workflow system provides control of the process, and substantial productivity—thus it saves costs.

Administrative

A third type of workflow is sometimes listed, called administrative workflow. This is a cross between the adhoc and production. The flow is pre-defined, such as the steps required to place an order or approve an expense report. Sometimes the flow can be reviewed in advance or even altered. The delivery may be a blend between e-mail, and a custom delivery mechanism.

Horizontal vs. Vertical

Another useful segmentation of the workflow technology is sometimes called horizontal vs. vertical workflow. Horizontal workflow moves the work through the organization—from person to person, or to different departments or systems. Once the work gets to a point in the organization, there may be several steps that are performed. The vertical workflow directs the processing at each step. It may automatically invoke computer programs, enter key data, and may provide guidance for each step of the process—at least for beginners if not for everyone.

Embedded

In some cases the workflow process is part of the business application. As previously noted, the value of workflow may be so great for an application that the vendor of that application includes workflow, so that “everyone” has it. Many of the embedded workflow systems are simple, but optimized for that particular type of use so may be adequate.

Autonomous (stand-alone)

If users deal with many applications (as seems to be the normal case) then a separate stand-alone workflow application may be a better solution, rather than one built into an application. This separate system can be optimized for the total business requirement, rather than just one application. These separate systems support many business applications.

1.4 Layout of Thesis

The thesis has been organized to study the concept of a workflow management system and then to design and implement a workflow for Computer Engineering Department, Delhi College Of Engineering.

(a) Chapter 1 gives an introduction about the basic concepts of a Workflow Management System as also the Workflow technology. It also tells about the benefits derived from using such a system.

(b) Chapter 2 covers the Literature Review. The Workflow Reference Model given by WFMC is cited along with the current research effort, and also the various existing Workflow Systems.

(c) Chapter 3 presents the design and analysis of a workflow process using the UML tools. A brief analysis of Workflow Patterns is given to familiarize with the basic building blocks for process definition. The problem definition and UML design for “Design and Development of a Workflow Management System for DCE” is formalized.

(d) Chapter 4 presents the full implementation details followed in this work to develop the required Workflow Management System and the results are also presented.

(e) Chapter 5 highlights the Conclusion and Future of this work.

1.5 Purpose of the Thesis

The purpose of this Thesis work is to conceptualize, design and develop the WorkFlow Management System in the Computer Engineering Department and implement a framework, which facilitates workflow process design. This framework should provide features in consonance with the current technology levels available in enterprise process management, adapted to a professional institute. The framework needed to be open to modifications, extension and upgrade for process implementation as per the user requirements. This workflow management system should include the following functional and operational mandates:

(a) Process Modeling: The system must support modeling and automation of processes across multiple process categories.

(b) Process Interoperability: The system must support Communication and interoperability between separate Workflow applications.

(c) Infrastructure Leverage: The system must provide complete and seamless support for existing IT network infrastructures, leveraging all available system services and complying wherever with existing standards, policies and procedures.

(d) Internet and Messaging-based Interaction: The system must support the use of the Internet and standard store and forward messaging services.

Chapter 2

Literature Review

Since this is a thesis about workflow management, this chapter delineates what workflow is and what it is not. Workflow shares characteristics with many other systems. This chapter also discusses the characteristics that set workflow apart from them. Understanding the differences helps developers adapt techniques developed for these systems to workflow management.

2.1 Workflow Technology

Workflow technology has been used for decades. In the 1970s, focus was on procedures for Office Information Systems [1, 2]. Research during the 1980s placed more emphasis on process models as Ellis and Nutt [3] worked on Petri net models. Workflow expanded into fields like office automation and document imaging. But the technology didn't meet the expectations of business users. Unlike the rigid workflow solutions of that time, environments for situated work offered a less restrictive alternative and captured some of the momentum. Researchers realized that the success of workflow technology was limited by their narrow perspective. Therefore, they reconsidered workflow as a multidisciplinary endeavor, located at the intersection of different

areas of computer, management and social sciences. This broad perspective contributed to the return of interest in workflow technology in the 1990s. During the last few years workflow has been the focus of intense activity in terms of products, standards and research work as discussed by Mohan [4].

2.1.1 Definitions

Since its early days, researchers have proposed various definitions for workflow. Many of these definitions define workflow within a single domain. Probably the most notable and widely documented domain is Business Process Reengineering (BPR/BPM) as discussed by David Hollingsworth [5, 6]. Later workflow technology was widely deployed in banking, accounting, manufacturing, brokerage, insurance, healthcare, telecommunications, customer service, and engineering, and more recently, scientific experiments. Therefore, the following definition is preferred because it doesn't depend on a particular domain.

"***A workflow represents the operational aspects of a work procedure, the structure of tasks, the applications and humans that perform them; the order of task invocation; task synchronization and the information flow to support the tasks; and the tracking and reporting mechanisms that measure and control the tasks*.**"

The workflow literature refers to the software that enables people to define and execute workflows as WorkFlow Management Systems (WFMS). A workflow management system automates processes by managing jobs and resources. The WFMC [6] provides the following definition for a workflow management system:

"***A system that completely defines, manages and executes workflows" through the execution of software whose order of execution is driven by a computer representation of the workflow logic. A workflow management system automates the process logic***."

Humans and software applications (processing entities) perform workflow tasks, thus implementing the task logic. This separation of process and task logic allows workflow users to modify one without affecting the other. It also promotes software reuse and the integration of heterogeneous software applications.

2.1.2 Workflow and Process Automation

Workflow Management Coalition (WFMC) [5] recommends a two tier model for processes that are automated with workflow. The coalition recommends a flow tier that automates the process logic and a work tier that contains the process activities as shown in Figure No. 2.1. Process technology has become ubiquitous and as per Sheth et all [7]. Workflow users who design and optimize these processes work with the high-level process descriptions on the flow tier.

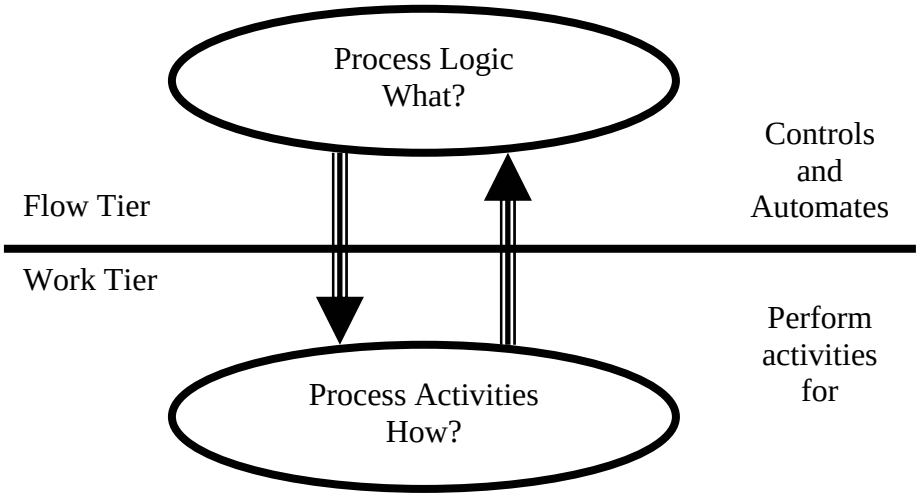

Figure No. 2.1 The Two Tier Model of the Process Logic and Activities

2.2 The Evolution of Workflow

Many types of products in the IT market have supported aspects of workflow functionality for a number of years, yet it is only comparatively recently that its importance has been recognized in its own right. The evolution of workflow as a technology has thus encompassed a number of different product areas as highlighted by David Hollingsworth [6]. These processes span Image and document management, Electronic mail and directories, Groupware applications, Transaction based applications and Business process Re-engineering.

2.3 Workflow Features

In a recent study Sheth et all [7] estimates the number of available workflow products between 200 and 300. Typically a new workflow system differentiates itself from others by offering features that are not available in other systems but the majority of these systems share a small set of common features. The minimal common set of features aim at providing the flow independence, domain independence, process logic monitoring and history log and provisions for manual intervention.

2.3.1 Flow-Independence

There are many different concerns in a piece of software. Data management and user interface represent two of the aspects that many applications have to deal with. It is required to separate the issues of low-level data management and user interfaces. The application manages its data through the mechanisms provided by the database system. Therefore, applications that rely on database technology for data management become data-independent. Likewise, user interface frameworks handle the issues of user interfaces. They enable developers to build interface-independent applications. Workflow enables developers to separate the flow between an application's components/modules/objects from the application (i.e., the process). Flow-dependent software implements application-specific components and the flow between them.

Most applications fall into this category since usually the underlying process emerges as the application evolves. Software developers implement the process models within their applications with workflow technology. Since application components have no knowledge of the sequencing of activities and their interdependencies, changing the process doesn't affect them. Thus, workflow applications become flow-independent. Additionally, workflow technology allows developers to use workflow-specific features that otherwise would be too expensive to handcraft every time they build a new application.

2.3.2 Domain-Independence

The partitioning typical of flow-independent applications (Figure No. 2.1) keeps the workflow outside the application domain. Thus applying workflow to a particular application domain requires providing components that perform domain-specific work. This characteristic makes workflow technology applicable to a large number of application domains and Georgakopoulos et all [8] discuss good examples from the telecommunications industry. Workflow implementations that focus on vertical markets discussed by Changengine [9] targets business administration and METEOR2 [10] provides workflow solutions for the health care industry etc. A wide range of application domains can benefit from current workflow technology.

2.3.3 Monitoring and History

The separation of process logic from the application components as proposed by WFMC [5] enables workflow to tap into the process level and collect information about its execution. Workflow systems provide this data at run time and after the process is complete. A workflow monitor enables workflow users to examine this information at run time. Workflow management systems also record the state of the running processes as these unfold in time. Workflow history involves a persistent store and aims at providing an audit trail after the workflow completes. Some workflow systems use the logged information for recovery. Workflow designers use it for process analysis, where history information forms the basis for improving the process.

2.3.4 Manual Intervention

Workflow management systems ensure that at run time processes execute according to their definition. Under exceptional circumstances, the workflow user needs to override the process definition and manually change the course of the process. Han et all [11] discuss feature that enable workflow systems to handle exceptions and unique situations. Early workflow systems didn't provide this functionality. Current workflow systems aim at providing various degrees of flexibility. Consequently, applications that use workflow to implement processes allow their users to manually change running processes by simply leveraging this feature of the workflow management system.

2.4 Workflow Standards

Despite the fact that people have used workflow technology for over two decades, a few years ago no workflow standards were available. But as workflow expanded from image and document-routing to business reengineering to mainstream middleware for process automation, interoperability issues became important. Consequently, during the last few years the workflow community has been working on standardization. The first standardization effort dates from 1994. The Workflow Management Coalition (WFMC), an organization of workflow product vendors, researchers, and users founded in 1993, developed the Workflow Reference Model [6, 12]. Initially the reference model focused on defining programmatic interfaces to workflow engines, aiming at standardizing the following five interfaces:

(a) **Interface 1**: Defines a common format for the interchange of static process specifications.

(b) **Interface 2**: Enables workflow participants to control process execution and manipulate work items.

(c) **Interface 3**: Provides access to the workflow applications.

(d) **Interface 4**: Enables different workflow servers to interact with each other.

(e) **Interface 5**: Provides an entry point for administration and monitoring tools.

Consequently, the focus shifted from specifying APIs towards the specification of meta-models and abstract interfaces. The recently adopted OMG Workflow Management facility adapts the WFMC runtime standard to a business object execution environment. The OMG is currently working on finalizing the standardization of the workflow facility. Research efforts in the workflow domain have uncovered weaknesses in the implementations based on reference models proposed by the WFMC and the OMG [13]. Paul et al [14] highlights that the WFMC standard represents a necessary step in the direction of workflow system interoperability; the current version of the standard has significant weaknesses that limit its value in a heterogeneous, distributed environment. Most of the weaknesses in the standard stem from the monolithic nature of the workflow server, which impedes the flexibility and scalability of workflow systems.

2.5 Review of available Workflow System

The definition of workflow as the automation of a process is very general and there are very many ways to design and implement a workflow system [2, 15,]. At the simpler end of the scale, a business process can be built into documents, such as spreadsheets, with macros, which of course is not very flexible. At the other end of the scale, highly complex workflows can be implemented using dedicated commercial workflow tools such as that built into Oracle or BEA WebLogic.

2.5.1 Workflow systems based on Macros

Both Microsoft Excel and Word allow documents to have functionality embedded within them in the form of macros. Using Visual Basic for Applications (VBA) a static document can be extended to perform actions based on button clicks or values that are modified. Workflow systems can evolve from simple beginnings rather than being designed up front in a large analysis phase. If the system grows in an informal way with no upfront analysis of the process, it can become difficult to maintain Version control of spreadsheet templates can become a problem.

The quality and robustness of the system will depend on the level of programming skill of the people writing the macros. This is a very rudimentary implementation of workflow with no set norms and rules and is not a viable option for serious workflow usage.

2.5.2 Systems Based on Workflow Packages

2.5.2.1 Oracle Workflow and Oracle Workflow for Java (OW4J)

Oracle comes with a Business Process Management (BPM) tool called Oracle Workflow [16, 17], and is being evolved in Oracle 10g with the next generation of Java based tools in the form of Oracle Workflow for Java, or OW4J. Oracle Workflow supports business process modelling, execution and monitoring. A graphical user interface allows an analyst to model a business process as a flowchart. When the workflow is executed tasks can be assigned, email notifications sent and application code executed. The workflow definition itself is saved to a database and can be version controlled. Defining a workflow in this way gives greater flexibility than hard-coded solutions. Oracle Workflow for Java (OW4J) is the next generation of J2EE-based business process management solution. The user interface for defining workflows is found in JDeveloper, which is Oracle's development environment for Java developers. OW4J complements rather than replaces the Oracle workflow engine.

2.5.2.2 BEA WebLogic and Business Process Management

BEA WebLogic Platform [18] is a service-oriented architecture (SOA) consisting of an application server, JVM (Java Virtual Machine), enterprise portal and application development framework. BEA's tools offer a standards-based application infrastructure that provides customers with business flexibility. The Business Process Management functionality within the BEA WebLogic suite of applications provides a sophisticated workflow engine to help automate business processes. BPM is all about managing enterprise level business processes and incorporates the integration of heterogeneous systems, also known as EAI (Enterprise Application Integration). BEA WebLogic's Integration Studio allows specialists to design, execute and monitor complex processes spanning multiple systems and people. As with Oracle Workflow, a graphical user interface is provided in which a business process can be modeled as a flowchart. Then a workflow template is saved to a database such as Oracle or SQL Server. An

action might be to invoke an EJB (Enterprise Java Bean) method or send an XML message to another application. The process engine manages execution of the workflow. XML is used to represent data and JMS (Java Message Service) is used for communication with other workflows and applications.

2.5.2.3 Jboss jBPM – open source extensible workflow management system

JBoss [19] is an established open source company with a focus on Java based middleware. The JBoss application server is a very popular J2EE platform. JBoss jBPM is a workflow management system in which a process is defined formally in a specialist language called JBoss jBPM Process Definition Language (jPDL). jBPM is a workflow engine that sits in the middle of a system of enterprise applications allowing integration and coordination of the separate applications. Enterprise Application Integration (EAI) is a big topic in the industry at the moment, which is why JBoss, Oracle, BEA etc are all offer competing workflow based products to meet this need.

2.5.3 Web-based workflow systems and BPEL

Web Services (using SOAP, WSDL and XML) are now the standard for doing business over the web. The idea behind a web service is that organizations provide services that are available to be called programmatically over the web [15]. Extending the concept to the next level it has been realized that often a business process may involve multiple business partners offering web services that can be hooked up to define an end-to-end business process. Web services alone do not provide the technology to do this and so BEA, IBM and Microsoft got together in 2002 to come up with a new language called BPEL (Business Process Execution Language for Web Services). BPEL is an abstract, XML based process definition language for defining an end-to-end business process flow. It supports asynchronous transactions, flow control and compensating business logic. In order to execute a process a BPEL script is parsed by a BPEL engine. The script is a definition of the workflow and can be executed by any BPEL compliant engine. It is not platform or vendor specific.

2.5.4 Other high-level workflow engines

As well as the offerings from Oracle, BEA, JBoss etc, there are organizations that have developed their own workflow engines, which are designed as components that can be integrated into new and existing systems. Below are just four examples, but there are literally dozens of products, each offering a unique set of features and with varying levels of compliance with emerging standards.

2.5.4.1 Verity Liquid BPM

The Verity Liquid BPM [15] (formerly the Drala workflow engine from Drala Software Inc) is an embeddable Java component, which can handle hundreds of workflows simultaneously. The engine supports persistence of workflows to a relational database. Complementing the workflow engine is the Workflow Studio, a visual environment for modelling workflows. The Workflow Manager provides monitoring and management services. The BPEL Orchestrator provides tools to build and automate processes using web services based on the BPEL4WS (Business Process Execution Language for Web Services) standard.

2.5.4.2 Enhydra Shark

Enhydra Shark [20] is an open source extendable workflow engine based on Workflow Management Coalition (WFMC) specifications using XPDL (XML Process Definition Language). It is built using Java/XML technology. XPDL is a workflow definition language similar in concept to BPEL.

2.5.4.3 OpenSymphony OSWorkflow

OpenSymphony [21] is another open source project dedicated to J2EE enterprise components. OSWorkflow is a workflow engine, which the designers claim stands out from the crowd due to its very high flexibility. This might be of interest to someone who feels that many of the high-level workflow engines lack flexibility, but who doesn't want to go completely down the road of writing everything from scratch. The recommended approach here is to write XML process descriptors "by hand" rather than in a graphical tool (though a graphical tool has now been included).

2.4.5.4 OpenWF

OpenWF [22] is a workflow management solution from OpenWF.com. It is developed using Microsoft technology including .NET, APS.NET, SQL Server. On offer are a workflow graphic designer, and a workflow engine. A web application interface is provided so no special client software is needed.

The various Workflow Engines reviewed in this chapter, along with their web addresses are attached as Annexure A.

2.6 Review of OpenWFE

Open WFE is an Open source Workflow Engine and is the choice for the implementation of the Workflow for this thesis. It is also a complete workflow management system [23] and is designed from scratch in conformity with WFMC specifications. This engine has the capability to use python base APRE invocation that extends its capability to interface with a variety of applications in the work tier. The WFMS uses the XML process definition language, which generated the flexibility in modification and implementation of new processes quickly and efficiently. This feature also provides interoperability with standard workflow definition languages like XPDL as recommended by WFMC. Since the workflow system is an open source, it gives that flexibility to evolve based on the requirement of the specific implementation.

2.7 UML and WorkFlow

UML activity diagrams are intended to model both computational and organisational processes [24]. However, if activity diagrams are to succeed as a standard in the area of organisational process modeling, they should compare favorably to the languages currently used for this purpose, that is, those supported by existing Workflow Management Systems (WFMS). Many of the workflow patterns offered by these languages are documented in [25, 26].

Chapter 3

Problem Definition and Development of WFMS

3.1 Workflow Management System

A typical workflow management system is standardized by WFMC and is depicted in Figure No. 3.1. It essentially has five interfaces that interact with the system engine to implement a WFMS.

Interface 1

This interface defines a common format for the interchange of static process specifications. Through this interface process definition is implemented. Various languages may be used for process definition but the current standards are XPDL (XML Process Definition Language) and XML (Extensible Markup Language) based.

Interface 2

This interface is the front-end of the system what a user will see. It enables workflow participants to initiate, control process execution and manipulate work items.

Interface 3

This interface provides access to the workflow applications. These applications are generally the helper operating system provided facilities like email, database, word processing etc that are required by the flow logic to propagate work items as per the process definition.

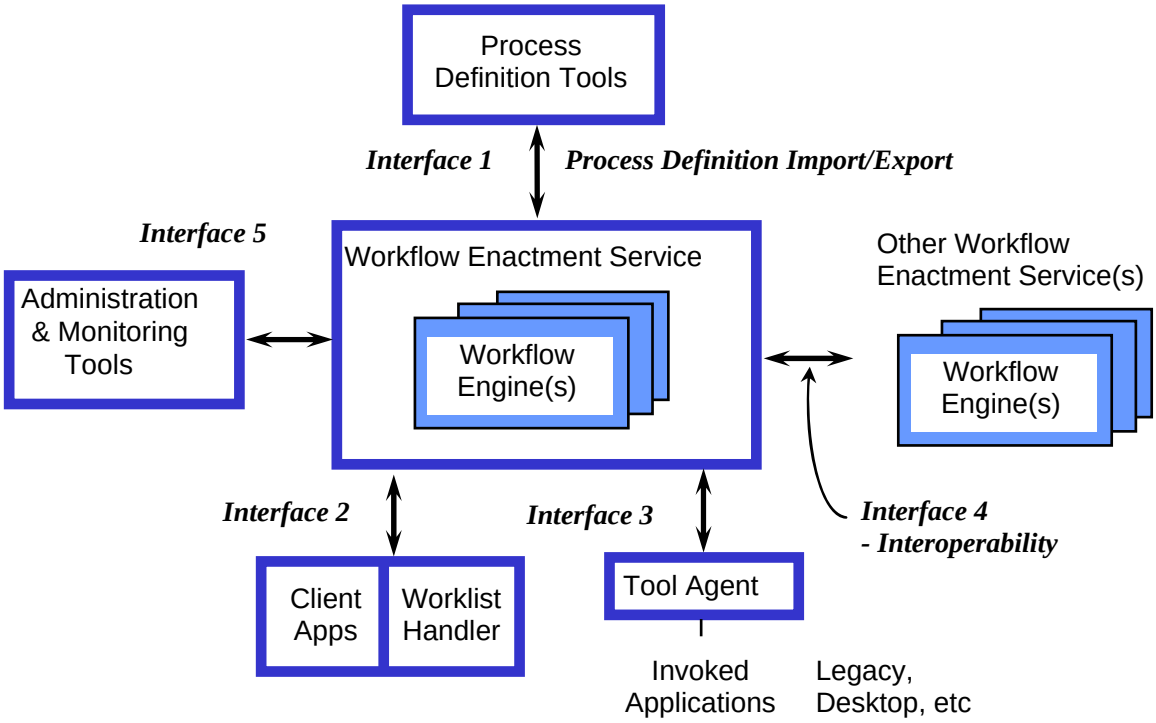

Figure No. 3.1 Workflow Reference Model –Components and Interfaces

Interface 4

This interface implements the scalability of the WFMS. It enables different workflow servers to interact with each other. These servers may be functionally or geographically distributed. Examples are like marketing department workflow process initiates a process in the shipping department to dispatch an item.

Interface 5

This interface is the entry point for administration and monitoring of the WFMS. This is used to control users, monitor work items etc when the workflow server is operative and dynamically control the server, if required.

3.2 UML Activity Diagrams

UML activity diagrams are intended to model both computational and organisational processes. Thus the UML diagrams logically form the first step in modeling of a business process corresponding to any application of workflow. The UML activity diagram for a process is the basic of designing and modeling an organisational process.

States and transitions

UML activity diagrams are special cases of UML state diagrams, which in turn are graphical representations of state machines. State machines are transition systems whose arcs are labeled by ECA (Event- Condition-Action) rules. The occurrence of an event results in a transition if (i) the machine is in the source state of the transition, (ii) the type of the event occurrence matches the event description of the transition, and (iii) the condition of the transition holds. The event (also called trigger), condition (also called guard), and action parts of a transition are all optional. A transition without an event is said to be triggerless. Triggerless transitions are enabled when the action or activity attached to their source state is completed. A state can contain an entire state machine within it, leading to the concept of compound state.

In the context of workflow specification, the strong points of activity diagrams with respect to alternative languages provided by commercial WFMS are essentially the following:

(a) They support signal sending and receiving at the conceptual level.

(b) They support both waiting states and processing states.

(c) They provide a seamless mechanism for decomposing an activity specification into sub-activities. The combination of this decomposition capability with signal sending yields a powerful approach to handling activity interruptions.

However, activity diagrams exhibit the following drawbacks:

(a) Some of their constructs lack a precise syntax and semantics. For instance, the well-formedness rules linking forks with joins are not fully defined, nor are the concepts of dynamic invocation and deferred events, among others.

(b) They do not fully capture important kinds of synchronization such as the discriminator and the N-out-of-M join.

(c) They do not fully support the producer-consumer pattern with termination activity

3.3 Workflow Patterns

The primary task of a workflow management system is to enact case-driven business processes by allowing workflow models to be specified, executed, and monitored. Workflow process definitions (workflow schemas) are defined to specify which activities need to be executed and in what order (i.e. the routing or control). An elementary activity is an atomic piece of work. Workflow process definitions are instantiated for specific cases. Since a case is an instantiation of a process definition, it corresponds to the execution of concrete work according to the specified routing. Activities are connected through transitions and we use the notion of a thread of execution control for concurrent executions in a workflow context. Activities are undertaken by roles, which define organizational entities, such as humans and devices. Control data are data introduced solely for workflow management purposes, e.g. variables introduced for, routing purposes. Production data are information objects (e.g. documents, forms, and tables) whose existence does not depend on workflow management. Elementary actions are performed by roles while executing an activity for a specific case, and are executed using applications.

Workflow patterns are typically realized in a specific language using one or more constructs available for this language. Sometimes workflow constructs available for a given language are not sufficient to realize a given pattern and workflow implementers have to resort to programming techniques such as event queuing, database triggers, etc to circumvent the limitations of a given workflow tool. A standard set of workflow patterns were studied by Van der Aalst, Hofstede, Kiepuszewski and Barros [25, 26] and are used to model any complex workflow process definition. These patterns have been accepted as standard patterns by WFMC and are detailed in following paragraphs. All the process design activities aim at synthesizing the major task into a combination of these patterns.

3.3.1 Basic Control Flow Patterns

These patterns closely match the definitions of elementary control concepts provided by the WFMC. These are:

(a) Pattern 1 (Sequence): The second activity occurs after the first one is completed.

(b) Pattern 2 (Parallel Split): Multiple activities start based on the completion of an activity.

(c) Pattern 3 (Synchronization): Activity is executed only after multiple prerequisite activities are completed.

The next two patterns are used to specify conditional routing. In contrast to parallel routing only one selected thread of control is activated.

(a) Pattern 4 (Exclusive Choice): This activity is executed if the result of a previous activity is exclusively tested true.

(b) Pattern 5 (Simple Merge): In this pattern the activity is executed if one of the split branches generates a result.

3.3.2 Advanced Branching and Synchronization Patterns

Here the focus will be on more advanced patterns for branching and synchronization. As opposed to the patterns in the previous section, these patterns do not have straightforward support in most workflow engines.

(a) Pattern 4 (Exclusive choice): assumes that exactly one of the alternatives is selected and executed.

(b) Pattern 6 (Multi-choice): A point in the workflow process where, based on a decision or workflow control data, a number of branches are chosen.

(c) Pattern 7 (Synchronizing Merge): A point in the workflow process where multiple paths converge into one single thread. If more than one path is taken, synchronization of the active threads needs to take place. If only one path is taken, the alternative branches should re converge without synchronization.

(d) Pattern 8 (Multi-merge): A point in a workflow process where two or more branches reconverge without synchronization. If more than one branch gets activated,

possibly concurrently, the activity following the merge is started for every activation of every incoming branch.

(e) Pattern 9 (Discriminator): The discriminator is a point in a workflow process that waits for one of the incoming branches to complete before activating the subsequent activity. Once all incoming branches have been triggered, it resets itself so that it can be triggered again Example: To improve query response time, a complex search is sent to two different databases over the Internet. The first one that comes up with the result should proceed the flow. The second result is ignored.

3.3.3 Structural Patterns

Different workflow management systems impose different restrictions on their workflow models. These restrictions are not always natural from a modeling point of view and tend to restrict the specification freedom of the business analyst. As a result, business analysts either have to conform to the restrictions of the workflow language from the start, or they model their problems freely and transform the resulting specifications afterwards.

(a) Pattern 10 (Arbitrary Cycles): A point in a workflow process where one or more activities can be done repeatedly.

(b) Pattern 11 (Implicit Termination): A given sub process should be terminated when there is nothing else to be done.

3.3.4 Patterns involving Multiple Instances

The patterns in this subsection involve a phenomenon that is referred to as multiple instances. From a theoretical point of view the concept is relatively simple and corresponds to multiple threads of execution referring to a shared definition. From a practical point of view it means that an activity in a workflow graph can have more than one running, active instance at the same time.

(c) Pattern 12 (Multiple Instances without Synchronization): Within the context of a single case (i.e., workflow instance) multiple instances of an activity can be created, i.e.,

there is a facility to spawn off new threads of control. Each of these threads of control is independent of other threads. Moreover, there is no need to synchronize these threads.

(d) Pattern 13 (Multiple Instances with a Priori Design Time Knowledge): For one process instance an activity is enabled multiple times. Once all instances are completed some other activity needs to be started.

(e) Pattern 14 (Multiple Instances with a Priori Runtime Knowledge): For one case an activity is enabled multiple times. The number of instances of a given activity for a given case varies and may depend on characteristics of the case or availability of resources, but is known at some stage during runtime, before the instances of that activity have to be created. Once all instances are completed some other activity needs to be started.

3.3.5 State-based Patterns

In real workflows, most workflow instances are in a state awaiting processing rather than being processed. Many computer scientists, however, seem to have a frame of mind, typically derived from programming, where the notion of state is interpreted in a narrower fashion and is essentially reduced to the concept of data.

(a) Pattern 16 (Deferred Choice): A point in the workflow process where one of several branches is chosen. In contrast to the XOR-split, the choice is not made explicitly (e.g. based on data or a decision) but several alternatives are offered to the environment.

(b) Pattern 17 (Interleaved Parallel Routing): A set of activities is executed in an arbitrary order: Each activity in the set is executed, the order is decided at run-time, and no two activities are executed at the same moment (i.e. no two activities are active for the same workflow instance at the same time).

3.3.6 Cancellation Patterns

These patterns are used to cancel and terminate the flow.

(a) Pattern 19 (Cancel Activity): An enabled activity is disabled, i.e. a thread waiting for the execution of an activity is removed.

(b) Pattern 20 (Cancel Case): A case, i.e. workflow instance, is removed completely (i.e., even if parts of the process are instantiated multiple times, all descendants are removed).

3.4 Problem Statement

The Computer Engineering Department of Delhi College Of Engineering caters to students at undergraduate, graduate and postgraduate levels in engineering. The administration requirements of each level of students are different with different informational needs. Currently a manual system of administration processes exists, which is inherently slow (physical progress of workitem), not amenable to a realtime status query and prone to loss of workitems. A system was required to automate the administration processes in the department so that the interacting faculty and students could access and work the specific admin chores in an efficient and effective manner.

3.4.1 Problem Definition

The problem taken up in this thesis is to develop a workflow management system for the fulfillment of the above mentioned functional and operational goals, so as to provide a framework for further implementation of various workflow processes based upon the departmental requirements.

The first step in this direction was to study the various administrative activities that are typically carried out in the computer engineering department and based on these the Use Case diagrams were created using Unified Modeling Language. Rational Rose software was used for preparing this representation of the actors, roles and their relationships.

3.4.2 DCE Course Registration Problem

At the beginning of each semester, students may request a course catalog containing a list of course offerings for the semester and the faculty. Information about each course, such as Professor, department, and prerequisites will be included through a web based interface to help students make informed decisions. Similarly the research interests of each professor are also made available through web based database browsing facility.

Once the student has selected the courses or the thesis topic he wants to register for he would initiate a workflow for the formal approval. Once the registration process is completed for a student, the registration system sends information to the billing system so that the student can be billed for the semester. Once the workflow for the registration is completed the system should update the database so that the Professors and students can access the relevant information through a web based interface.

The administrator is responsible for the generation of course catalog for a semester and for the maintenance of all information about the curriculum, the thesis, the students and the professors needed by the system. Head of department is involved with the approval of thesis research topic as well as the guiding faculty.

3.4.3 Modeling

The Behavior of the system under development (i.e. the functionality to be provided by the system) is documented in the Use Case model that illustrates the system's intended functions (use cases), its surroundings (actors), and the relationships between the use cases and actors.

3.4.3.1 Actors in the DCE Course Registration System

Analysis of the registration problem revealed that five principal actors in the system were:

(a) **Student**- a person who wants to register for courses/Thesis.

(b) **Administrator**- The person who is responsible for and controls the whole registration system.

(c) **Professor**-a person who is certified to take courses and to guide Thesis

(d) **HOD**- a person who is authorized to approve research guide and arbitrate in case of a dispute over multiple acceptance of thesis topic.

(e) **Billing system**-the external system responsible for student billing.

3.4.3.2 Use Case diagram and Use Cases in the Course Registration System

The Use Case Diagram is a graphical view of some or all of the actors, Use Cases and their interactions identified for the system under development. The main Use Case diagram for the DCE Registration system is made using Rational Rose software and is as shown in Figure No.3.2.

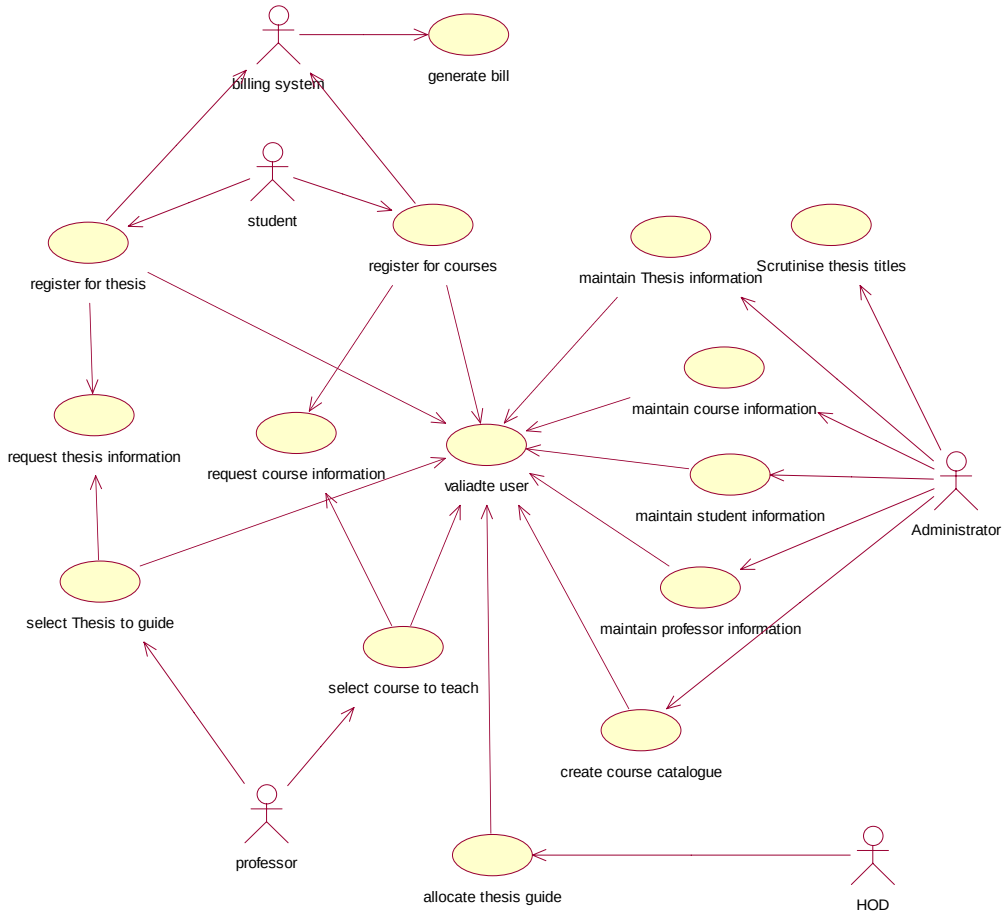

Figure No. 3.2 Use Case Diagram for DCE Registration System

Based on this Use Case Diagram, various use cases identified for the system along with their description were formulated. The Use Cases descriptions with associated contents are given in Table Nos. 3.1 through 3.12.

Primary Actor	A Student
Description	This allows the student to register for courses.
Flow of events Basic Flow Alternate Flow	The student logs in and enters the password and after viewing the Course Catalog registers for the desired courses. If the password is invalid alternate flow is started.
Special requirements	No special requirements
Preconditions	The Request Course catalog use case must execute before this use case begins.
Post conditions	The generate bill use case should be executed after registration.

Table No.3.1: Use case1- Register for courses

Primary Actor	A Student
Description	This use case allows the student to register for Thesis. It allows student to submit a topic and name of the two guides for his proposed work.
Flow of events Basic Flow Alternate Flow	The student logs in and is validated. It allows student to submit a topic and name of the two guides for his proposed work. If password is invalid, alternate flow is started.
Special requirements	No special requirements.
Preconditions	The request for Thesis use case should be executed.
Post conditions	The Scrutinize Thesis Title use case should be executed after registration.

Table No.3.2: Use case2- Register for thesis

Primary Actor	A Professor
Description	This use case is started by the professor and it provides the capability to select courses to teach for a selected semester.
Flow of events Basic flow Alternate flow	The Professor logs on to the system and enters the password and performs the desired actions and the case ends successfully. If password is invalid the alternate flow is executed.
Special requirements	No Special requirements.
Preconditions	The Maintain Course information use case must execute before this use case begins.
Post conditions	The Create Course Catalog Use Case executes after this Use Case ends.

Table No.3.3: Use Case3- Select courses to teach

Primary Actor	A Professor
Description	The professor starts this use case and it provides the capability to select Thesis to guide.
Flow of events Basic flow Alternate flow	The Professor logs on to the system and enters the password and requests for the Thesis information and performs the desired actions and the case ends successfully. If password is invalid the alternate flow is executed.
Special requirements	No Special requirements.
Preconditions	The request Thesis information use case must execute before this use case begins.
Post conditions	Maintain Thesis Information Use Case executes after this Use Case ends Successfully

Table No.3.4: Use Case4- Select Thesis to guide

Primary Actor	A Student or a Professor
Description	The student/Professor requests for the course information after logging in.
Flow of events Basic flow Alternate Flow	The Actor logs in successfully and requests for the information to make their individual decisions. If password is invalid, alternate flow is executed.
Special requirements	No Special requirements.
Preconditions	The maintain course information should be executed before the request for information use case begins.
Post conditions	There are no Post conditions.

Table No.3.5: Use Case5- Request course information

Primary Actor	A Student or a Professor
Description	The student/Professor requests for the thesis information after logging in.
Flow of events Basic flow Alternate Flow	The Actor logs in successfully and requests for the information to make their individual decisions. If password is invalid, alternate flow is executed.
Special requirements	No Special requirements.
Preconditions	The maintain Thesis information should be executed before the request for information use case begins.
Post conditions	There are no Post conditions.

Table No.3.6: Use Case6- Request Thesis information

Primary Actor	An Administrator
Description	The Administrator maintains updated information based upon the information received by him.
Flow of events Basic Flow Alternate Flow	The Administrator logs in successfully and updates the course information. If password is invalid, alternate flow is executed.
Special requirements	No special requirements
Preconditions	The register for Course use case is executed.
Post conditions	The create course catalog use case executes after this.

Table No.3.7: Use Case7 - Maintain course information

Primary Actor	An Administrator
Description	The Administrator maintains updated information based upon the information received by him.
Flow of events Basic Flow Alternate Flow	The Administrator logs in successfully and updates the student information. If password is invalid, alternate flow is executed.
Special requirements	No special requirements
Preconditions	The student wants to register for a course.
Post conditions	No Post conditions.

Table No.3.8: Use Case8- Maintain Student Information

Primary Actor	An Administrator
Description	The Administrator maintains updated information based upon the information received by him.
Flow of events Basic Flow Alternate Flow	The Administrator logs in successfully and updates the thesis information. If password is invalid, alternate flow is executed.
Special requirements	No special requirements
Preconditions	The Register for Thesis use case is executed.
Post conditions	No Post conditions.

Table No.3.9: Use Case 9- Maintain Thesis Information

Primary Actor	An Administrator
Description	The Administrator maintains updated information based upon the information received by him.
Flow of events Basic Flow Alternate Flow	The Administrator logs in successfully and updates the Professor information. If password is invalid, alternate flow is executed.
Special requirements	No special requirements
Preconditions	No preconditions.
Post conditions	No Post conditions.

Table No.3.10: Use Case 10- Maintain Professor Information

Primary Actor	An Administrator
Description	The Administrator is allowed to create a course catalog based upon the information received by him.
Flow of events Basic flow Alternate flow	The Administrator logs in successfully and creates the course catalog based on the professor and student information. If password is invalid, alternate flow is executed.
Special requirements	No special requirements
Preconditions	The register for course and select a course to teach use cases are executed before this use case executes.
Post conditions	No Post conditions.

Table No.3.11: Use case11- Create Course Catalog

Primary Actor	An Administrator
Description	The Administrator is allowed to Scrutinize the title submitted by the student for Thesis.
Flow of events Basic flow Alternate flow	The Administrator logs in successfully and Scrutinizes the title based upon Thesis information. If password is invalid, alternate flow is executed.
Special requirements	No special requirements
Preconditions	The register for Thesis use cases is executed before this use case executes.
Post conditions	Maintain Thesis information is executed after this use case ends successfully.

Table No.3.12: Use case12- Scrutinize Thesis Title

Primary Actor	HOD
Description	The HOD is allowed to allocate a thesis guide based on the information received by him.
Flow of events Basic flow Alternate flow	The HOD logs in successfully. If password is invalid, alternate flow is executed.
Special requirements	No special requirements
Preconditions	The Scrutinize thesis title and select Thesis to guide use Cases are executed before this use case executes.
Post conditions	Maintain Thesis information is executed for updation.

Table No.3.13: Use Case13- Allocate Thesis Guide

Primary Actor	Billing system
Description	The billing system generates a bill once the registration is done.
Flow of events Basic flow Alternate flow	The Billing system generates a bill whenever a successful registration happens. If password is invalid, alternate flow is executed.
Special requirements	No special requirements
Preconditions	The Register for Thesis or Register for Courses use cases are executed before this use case executes.
Post conditions	Maintain Thesis information or Maintain Course information use cases are executed for updation.

Table No.3.14: Use case14- Generate Bill

3.4.3.3 Activity diagrams

At this stage Activity Diagrams may be created to represent the dynamics of the system. They are flowcharts that are used to show the Workflow of the system; that is they show the flow of control from activity to activity in the system, parallel activities and any alternate paths through the flow. For the DCE course registration system two major activities are identified as registration for thesis and Create Catalog.

The activity diagram for registration of thesis is shown in Figure No. 3.3. This activity diagram shows the interaction between the four principal actors in the Use Case. This swim lane diagram forms the basis of moulding the specification for establishing the process definition in the workflow management system.

3.4.3.4 Designing using c++

Once the Use Case Diagram and the Activity Diagram had been developed, the next step was to go through the various stages of Rational Rose Software and in the end to generate the code in C++. It was observed that C++ language was not very amenable for the development of the Workflow Management System, as it was very difficult to capture the dynamic nature of the workflow and the proper routing of the workitems was very crude using the available language constructs. The overall structure of various classes used, based on the initial design was developed and is attached as Annexure B. On observing the various problems in using the said software, the decision to implement this work in C++ was not followed beyond the designing stage. In order to fully provide the functionality of the required nature as per the Use Case and Activity Diagrams, the choice had to be made of a workflow Engine that was latest and readily available for use, so as to be able to demonstrate the complete working WFMS and such an Engine was not compatible with C++. The various choices made and the subsequent building up of the WFMS along with the full implementation details and results is given in the following chapter.

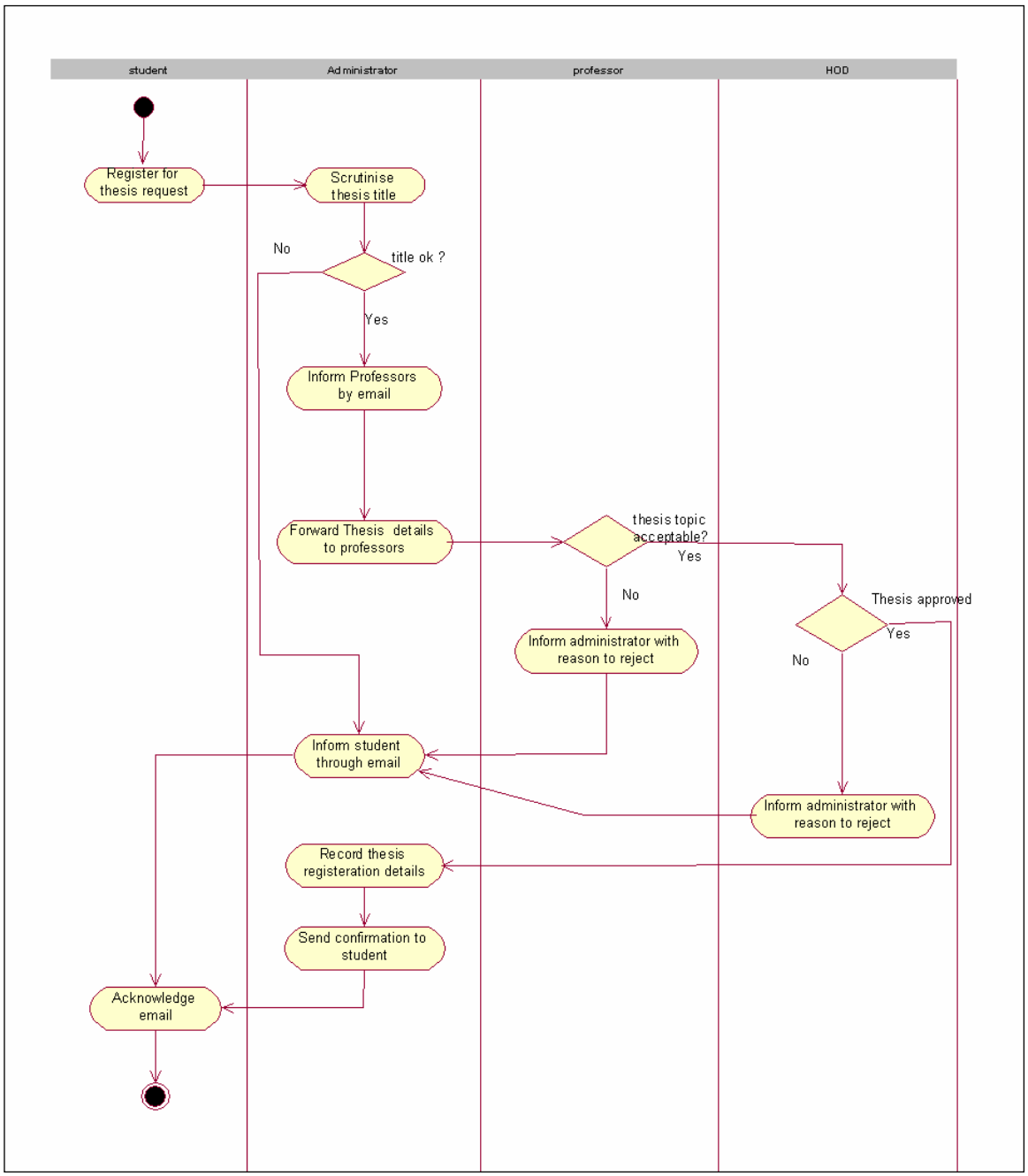

Figure No. 3.3 Activity Diagram for Thesis Topic Registration

Chapter 4

Workflow Management System Implementation

4.1 The Workflow

The activity diagrams conceptualized and developed for the problem in question depict the flow of information, the actions by individual actors on the workitems and dynamic routing of these work items based on actor response. A workflow management system is now needed for capturing this dynamic flow of information and interactions with the environment.

The first step in this direction was to establish the structure of the workflow management system. Recognized structure for a workflow management systems proposed by WFMC (as discussed in literature review) was the obvious starting point for concept building and implementation. The concept implementation involved a sequence of decisions and selection issues for the software components, which would shape the complete Workflow system as a single entity.

The Operating system of choice was the Linux Fedora Core 3. This was selected since it is an Open Source OS, extendable and upgradeable under GNU license and as such is readily available

for use and has no limitations of cost and features usage and so it is suitable for the development effort whose results are open to modifications and extensions.

The next part of the problem was the selection of participating services and software tools. For this OpenWFE 1.5.1 was selected. It is the only open source beta software available, which in itself is a full workflow management development system and comprises of the basic five components in consonance with the WFMC recommended structure. These components are:

Workflow Engine

The heart of the system, which is workflow enactment and execution environment, is built around JAVA. It requires JDK from Sun Microsystems which is again open source software. This engine works based on JAVA JDK1.4.2, and requires a POSIX compliant OS, which was selected to be Fedora Core 3 for this work.

Worklist

The worklist is a set of stores for workitems. Workitems are emitted by the engine upon encountering a Participant Expression. A worklist stores workitems and lets participants retrieve and use them. (Interface 2)

Webclient

This is also called the webappserver. The webclient is in fact a JAVA based web application devised to interact with the worklist. (Interface 2 and Interface 5)

Automatic Participant Runtime Environment

The APRE is a carrier for agent implemented either in java or in jython. This facility enables the engine to use the services of the applications residing on the host system. For example there is an agent named ’EmailNotificationAgent’ that is implemented in jython and that does what its name stands for. (Interface 3)

Droflo

Droflo is a web based workflow design tool. It means that you can build / edit workflows with your web browser and then save their XML process definition. OpenWFE uses its own process definition language, which is very powerful, flexible and which is extensible. (Interface 1)

4.2 Requirements of the WFMS

The system should provide skeletal infrastructure to define workflows, route workitems as per the flow definition. The student actor should be able to initiate flows based on his specific requirements and monitor its progress. The participants should be able to retrieve the pending task items, act upon them and handover to the WFMS. WFMS should route the work items as per the defined flows to the next participant. Administrator should be able to define new workflows and extend existing ones and view the flow across the complete process.

4.3 Scope

The scope of this work was to set up a WFMS Framework that was amenable to extensions and up-gradations. In doing so, various software were selected and integrated to provide a seamless support for existing IT network infrastructures, leveraging all available system services and complying with existing standards, policies and procedures. The implementation required to demonstrate a Working Model of this WFMS.

4.4 The Context

Within the scope of the thesis work, in order to demonstrate a working model of WFMS the scenario of thesis registration process, as conceptualized in the previous chapter, was chosen for implementation. A typical administration process in Computer Engineering Department at Delhi

College of Engineering involves the registration of Projects/Thesis by students at different levels and approval and allotment of faculty as Thesis guides. This process was implemented as a pilot process during this work and it essentially demonstrates the complete functionality of the WFMS.

4.4.1 Workflow Process Definition

A final year B.E. / M.E. student who wants to submit a topic for the Project/Thesis Work Logs on to the Workflow Management system and selects the link for 'Register for Thesis'. As a result the WFMS serves him/her the form to fill up the necessary details required as per the departmental Administration Policies for registering the choice. The student fills up a registration form with certain Personal information like the enrollment number and also the level i.e. whether the registration is for Graduate or for Post Graduate Thesis Work. The student then fills up the fields for the Project Title and also the choices for the names of two guides for the same. On submission the WFMS forwards it to the scrutinizer who will check for the validity of the student Personal information and if all is correct, validation of the title is done against an available database of all project Work undertaken by the Department so far. After validation WFMS duplicates the form and routes it to the next participant in the hierarchy i.e. the two guides for required action. The Guides shall find the Pending Workitems whenever they logon to the system. The guides can approve or disapprove the student request and also they can annotate the form and submit it to the WFMS. At any instant, the student can view the progress of his Workitem in the Workflow Management System. The form is then routed to the Head of the Department who can accept or reject the form. Based on his response the system will send a mail to the selected Guide and also intimation to the concerned student. In Addition the automatic updation of the database is done by the WFMS.

4.4.2 Workflow Centric Activity Diagram

The process definition as proposed in the preceding paragraph was implemented into a workflow centric process diagram. This process diagram used the UML activity diagram as its basis of definition. The process diagram comprises of several sub-processes that are executed on the main

engine through interface 1. Each of these diagrams has an XML code corresponding to it that is also detailed along with the diagram. The principle flow process is modeled in Droflo application that represents the graphical equivalent of the process XML code. The process and sub-process diagrams are shown in Figure Nos. 4.1 through 4.8. The registration process consists of eight activity process diagrams bearing nomenclature "*regflow__5.1.xml*" to "*regflow__5.8.xml*". The first code file "*regflow__5.1.xml*" is used to generate a launch form.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<workflow-definition name="regflow" revision="5.1">
<description>This flow is initiated by student for registration of thesis project</description>

    <sequence>
        <set value="false" type="boolean" field="Approved Topic" />
        <participant filter="closed1" ref="Administrator" />
        <if>
            <equals field-value="Approved Topic" other-value="true" />
            <subprocess ref="mainEngine::http://192.168.0.1:7079/regflow__5.2.xml" forget="true" />
            <subprocess ref="mainEngine::http://192.168.0.1:7079/regflow__5.3.xml" forget="true" />
        </if>
    </sequence>

<filter-definition
    name="closed1"
    type="closed"
    add="false"
    >
    <field regex="Approved Topic" permissions="rw" />
    <field regex="Proposed_Guide1" permissions="r" />
    <field regex="Proposed_Guide2" permissions="r" />
    <field regex="Thesis Topic" permissions="r" />
    <field regex="Students Name" permissions="r" />
    <field regex="Students Enrolment No" permissions="r" />
    <field regex="Email Id" permissions="r" />
    <field regex="Subject" permissions="r" />
</filter-definition>

</workflow-definition>
```

Listing 4.1 XML Code Listing for regflow__5.1.xml

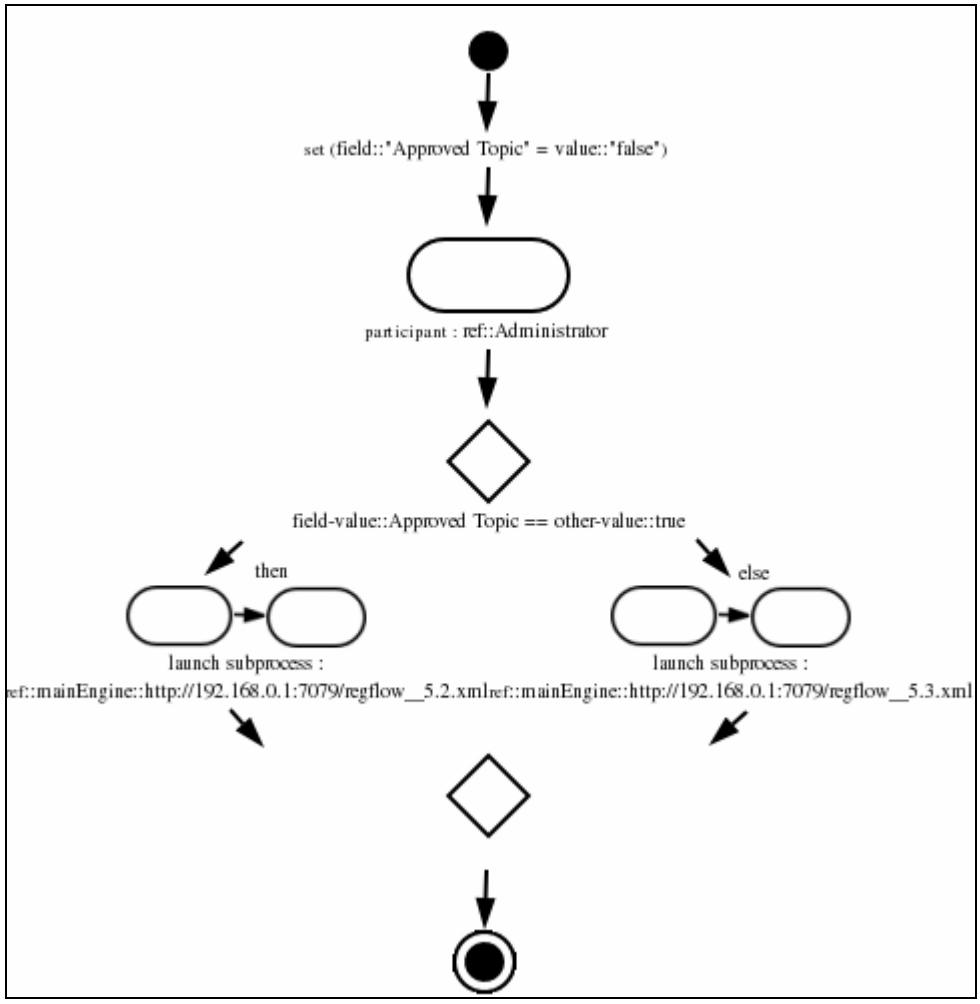

Figure No. 4.1 Droflo Equivalent Activity Diagram for regflow__5.1.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<workflow-definition name="regflow" revision="5.2">
<description>This flow Topic Request for approval to guides</description>
<sequence>
    <concurrence
        sync="generic"
        merge="last"
        merge-type="mix"
    >
    <sequence>
        <set field="Approved Guide1" value="false" type="boolean"/>
        <participant ref="Guide1" filter="closed2"/>
    </sequence>
    <sequence>
        <set field="Approved Guide2" value="false" type="boolean" />
        <participant ref="Guide2" filter="closed3"/>
    </sequence>
    </concurrence>
    <if>
        <and>
            <equals field-value="Approved Guide1" other-value="true" />
            <equals field-value="Approved Guide2" other-value="true" />
        </and>
        <!-- then -->
            <subprocess ref="mainEngine::http://localhost:7079/regflow__5.4.xml" forget="true" />
    </if>
    <if>
        <and>
            <equals field-value="Approved Guide1" other-value="false" />
            <equals field-value="Approved Guide2" other-value="false" />
        </and>
        <!-- then -->
            <subprocess ref="mainEngine::http://localhost:7079/regflow__5.5.xml" forget="true" />
    </if>
    <if>
        <or>
            <and>
                <equals field-value="Approved Guide1" other-value="true" />
                <equals field-value="Approved Guide2" other-value="false" />
            </and>
            <and>
                <equals field-value="Approved Guide1" other-value="false" />
                <equals field-value="Approved Guide2" other-value="true" />
            </and>
        </or>
        <!-- then -->
            <subprocess ref="mainEngine::http://localhost:7079/regflow__5.6.xml" forget="true" />
    </if>
</sequence>
<filter-definition
        name="closed2"
        type="closed"
        add="false"
    >
    <field regex="Approved Guide1" permissions="rw" />
```

```
    <field regex="Thesis for BE / ME" permissions="r" />
    <field regex="Students Name" permissions="r" />
    <field regex="Students Enrolment No" permissions="r" />
    <field regex="Thesis Topic" permissions="r" />
    <field regex="Email Id" permissions="r" />
    <field regex="Subject" permissions="r" />
</filter-definition>
<filter-definition
        name="closed3"
        type="closed"
        add="false"        >
    <field regex="Approved Guide2" permissions="rw" />
    <field regex="Thesis for BE / ME" permissions="r" />
    <field regex="Students Name" permissions="r" />
    <field regex="Students Enrolment No" permissions="r" />
    <field regex="Thesis Topic" permissions="r" />
    <field regex="Email Id" permissions="r" />
    <field regex="Subject" permissions="r" />
</filter-definition>
</workflow-definition>
```

Listing 4.2 XML Code Listing for regflow__5.2.xml

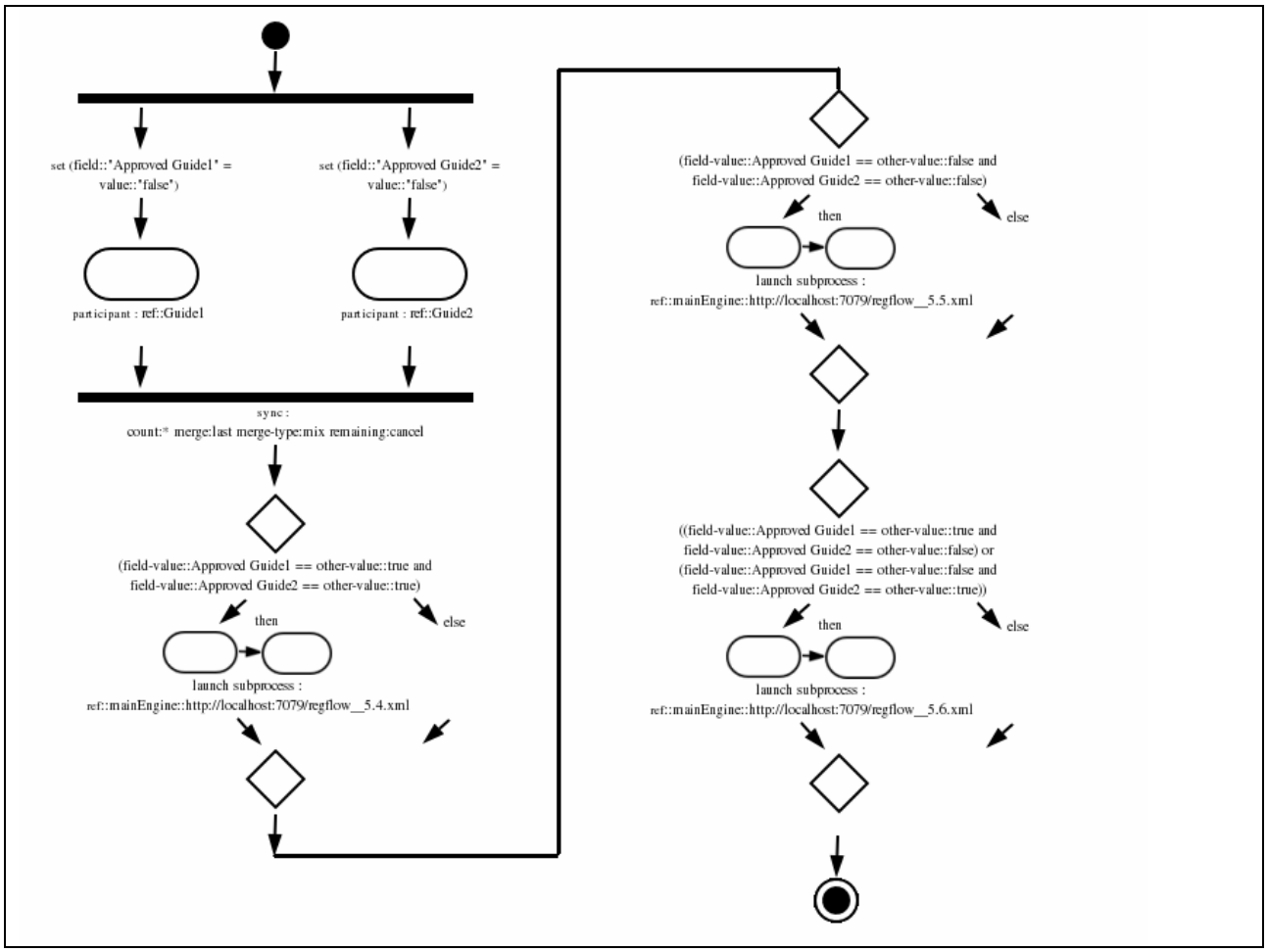

Figure No.4.2 Droflo Equivalent Activity Diagram for regflow__5.2.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<workflow-definition name="regflow" revision="5.3">
<description>This flow sends email to student about rejection of thesis topic</description>
<sequence>
    <set field-value="Email Id" field="__emailTarget__" />
    <set field-value="Subject" field="__emailSubject__" />
    <set value="Your Thesis topic has been rejected, Please resubmit the topic." field="__emailText__" />
    <participant ref="email-notif-agent" description="Thesis topic not accepted" />
    <set value="${field:Students Name}  ${field:Students Enrolment No}" field="__subject__" />
    <participant ref="Thesis" filter="readall" />
</sequence>
<filter-definition
        name="readall"
        type="closed"
        add="false"
    >
        <field regex=".*" permissions="r" />
</filter-definition>
</workflow-definition>
```

Listing 4.3 XML Code Listing for regflow__5.3.xml

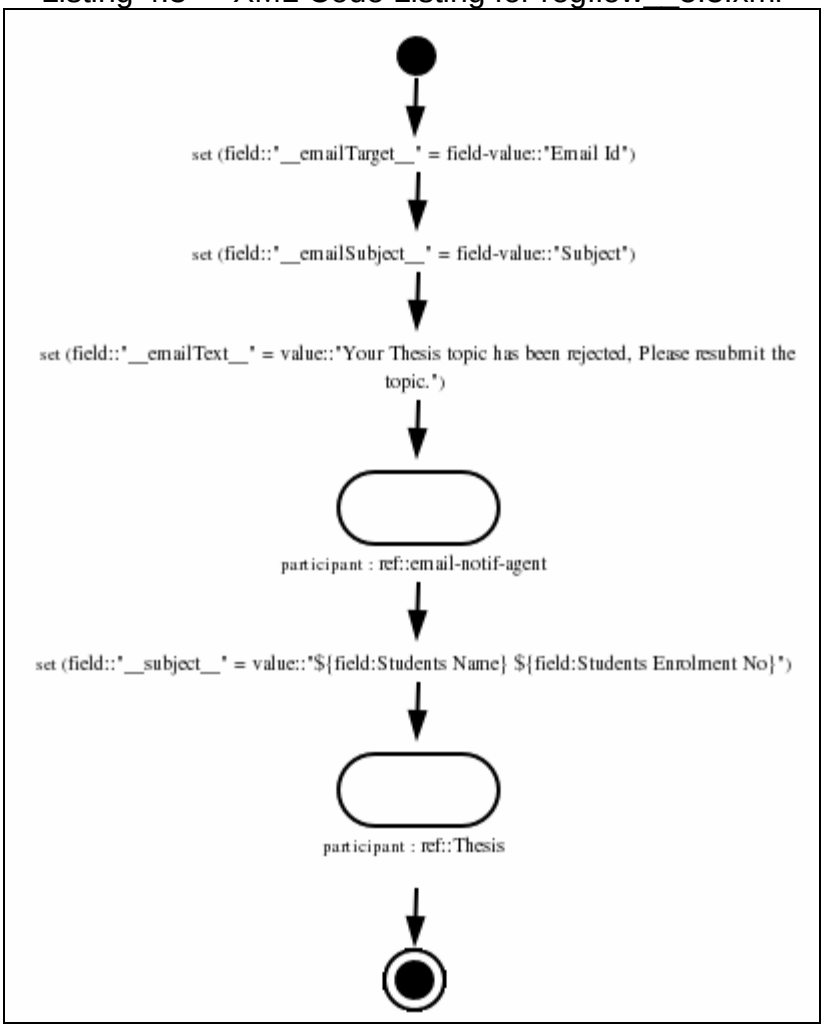

Figure No.4.3 Droflo Equivalent Activity Diagram for regflow__5.3.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<workflow-definition name="regflow" revision="5.4">
<description>This flow Requests HOD to arbitrate the allotment of Guide</description>

<sequence>
      <set field="HOD's Attention" value="Both guides have approved subject please allocate  one"/>
      <set field="Allocated Guide" value="Fill in the Name of approved Guide"/>
      <set field="Approval of HOD" value="false" type="boolean" />
      <participant ref="HOD" filter="closed2"/>
      <set field="Administrator's Attention" value="Both guides had approved HOD arbitrated and selected one"/>
      <participant ref="Administrator" filter="readall"/>
      <set field-value="Email Id" field="__emailTarget__" />
      <set value="Approval of Thesis Topic" field="__emailSubject__" />
<set value="Your Thesis Topic: ${field:Thesis Topic}  has been approved. Your Allocated Guide is
${field:Allocated Guide}" field="Students Email Text" />
      <set field-value="Students Email Text" field="__emailText__" />
      <participant ref="email-notif-agent"  />
      <set value="${field:Allocated Guide}@ganit.home.net" field="__emailTarget__" />
      <set value="Guide duties for Thesis" field="__emailSubject__" />
<set value="You have been assigned as guide for Thesis Titled: ${field:Thesis Topic} for ${field:Thesis for BE /
ME} as earlier approved by you. The Student details are as follows: Student's Name: ${field:Students Name} ,
Student's Enrolment No:  ${field:Students Enrolment No} ,  Student's Email Id:   ${field:Email Id}" field="Guides
Email Text" />
      <set field-value="Guides Email Text" field="__emailText__" />
      <participant ref="email-notif-agent"  />
      <set value="${field:Students Name}  ${field:Students Enrolment No}" field="__subject__" />
      <participant ref="Archiver"  filter="readall" />
</sequence>

<filter-definition
            name="closed2"
            type="closed"
            add="false"
      >
            <field regex="Proposed_Guide1" permissions="r" />
            <field regex="Proposed_Guide2" permissions="r" />
            <field regex="Students Name" permissions="r" />
            <field regex="Students Enrolment No" permissions="r" />
            <field regex="Thesis Topic" permissions="r" />
            <field regex="HOD's Attention" permissions="r" />
            <field regex="Subject" permissions="r" />
            <field regex="Approval of HOD" permissions="rw" />
            <field regex="Allocated Guide" permissions="rw" />
</filter-definition>
<filter-definition
            name="readall"
            type="closed"
            add="false"
      >
            <field regex=".*" permissions="r" />
</filter-definition>
</workflow-definition>
```

Listing 4.4 XML Code Listing for regflow__5.4.xml

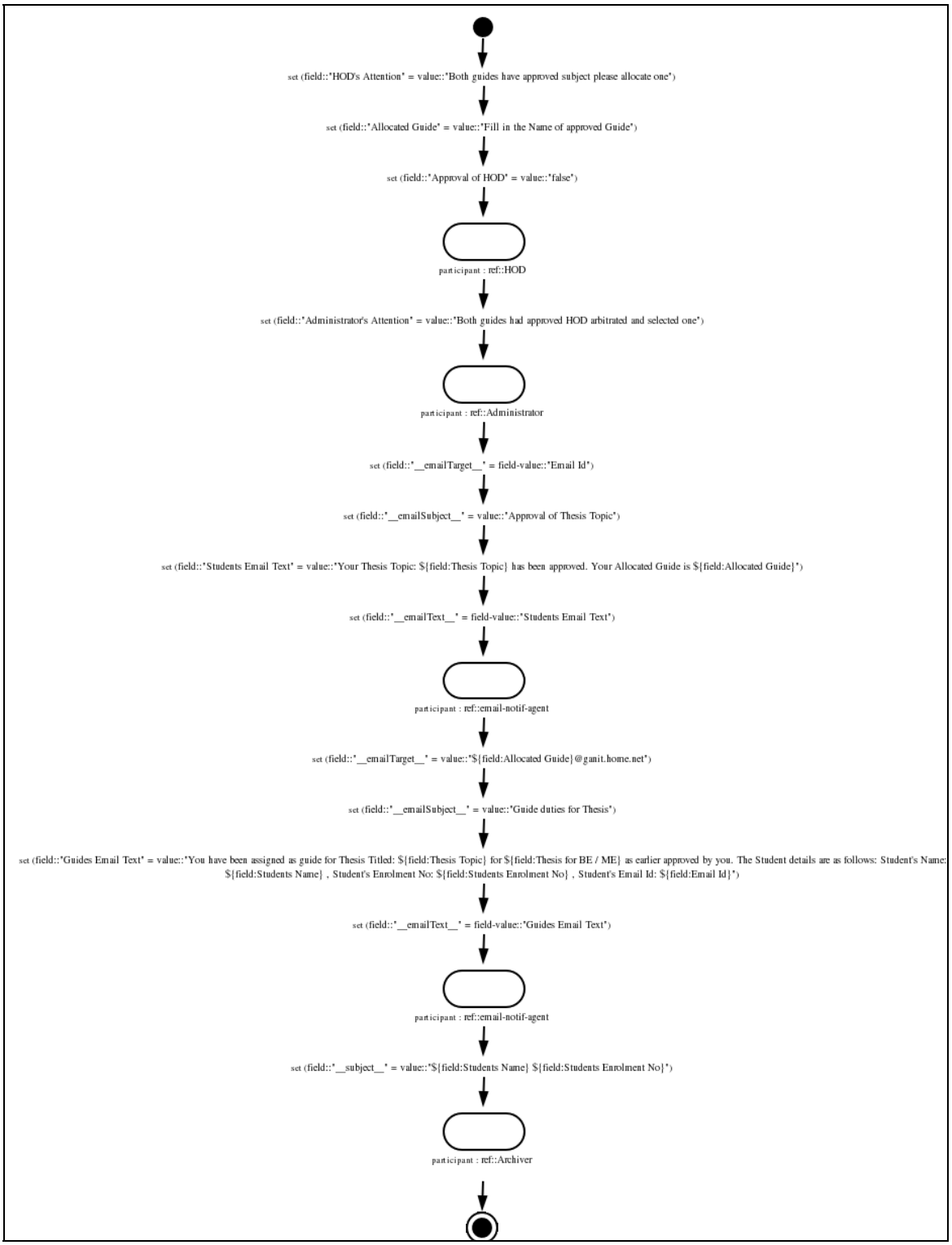

Figure No.4.4 Droflo Equivalent Activity Diagram for regflow__5.4.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<workflow-definition name="regflow" revision="5.5">
<description>This flow informs student that his thesis topic has not been accepted by guides</description>

<sequence>
     <set field-value="Email Id" field="__emailTarget__" />
     <set value="Both guides proposed by you in your initial request Prof ${field:Proposed_Guide1} and Prof
${field:Proposed_Guide1}  have denied guiding your thesis. Please resubmit request." field="__emailText__" />
     <set field-value="Subject" field="__emailSubject__" />
     <participant ref="email-notif-agent" description="Thesis topic not accepted" />
     <set value="${field:Students Name}  ${field:Students Enrolment No}" field="__subject__" />
     <participant ref="Thesis" filter="readall" />
</sequence>
<filter-definition
          name="readall"
          type="closed"
          add="false"
   >
          <field regex=".*" permissions="r" />
</filter-definition>
</workflow-definition>
```

Listing 4.5 XML Code Listing for regflow__5.5.xml

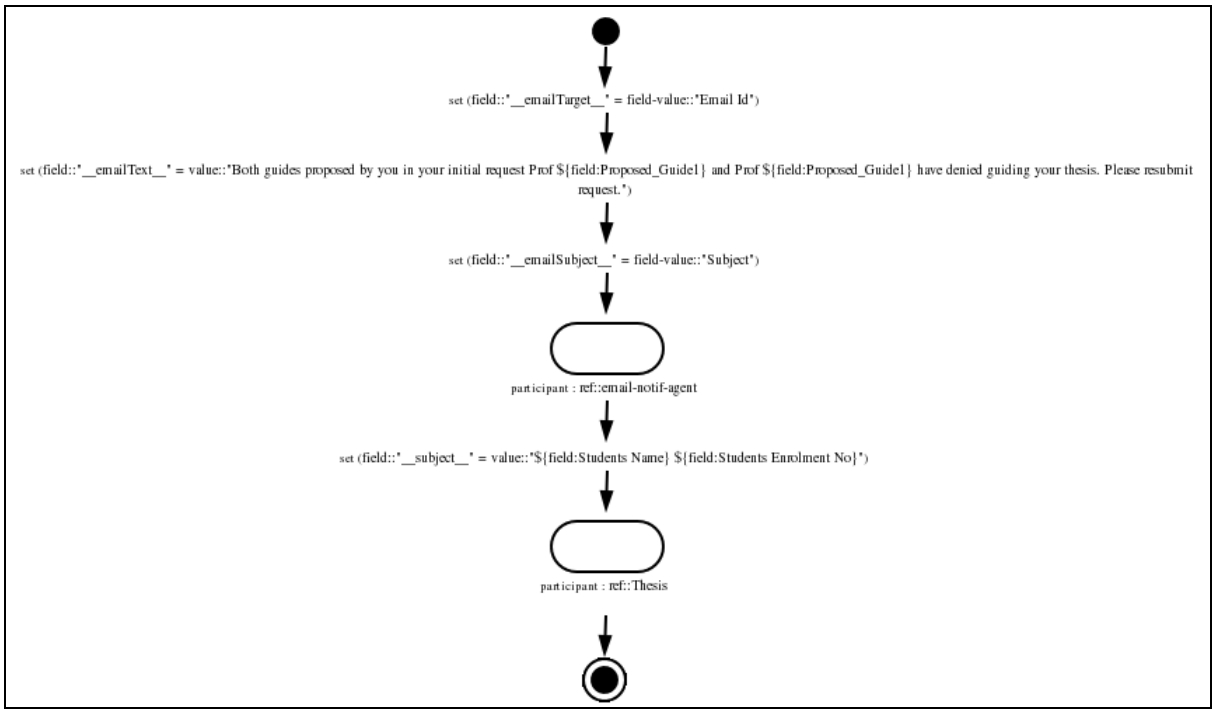

Figure No.4.5 Droflo Equivalent Activity Diagram for regflow__5.5.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<workflow-definition name="regflow" revision="5.6">
<description>This flow informs student about the allotment of Guide</description>

<sequence>
    <set field="HOD's Attention" value="One Guide has consented to Guide the thesis. Please Approve the name in
allocated coulmn"/>
    <if>
        <equals field-value="Approved Guide1" other-value="true" />
        <set field-value="Proposed_Guide1" field="Allocated Guide" />
        <set field-value="Proposed_Guide2" field="Allocated Guide" />
    </if>
    <set field="Approval of HOD" value="false" type="boolean" />
    <participant ref="HOD" filter="closed5"/>
    <if>
        <equals field-value="Approval of HOD" other-value="true" />
        <subprocess ref="mainEngine::http://localhost:7079/regflow__5.7.xml" forget="true" />
        <subprocess ref="mainEngine::http://localhost:7079/regflow__5.8.xml" forget="true" />
    </if>
</sequence>

<filter-definition
        name="closed5"
        type="closed"
        add="false"
    >
        <field regex="Students Name" permissions="r" />
        <field regex="Students Enrolment No" permissions="r" />
        <field regex="Thesis Topic" permissions="r" />
        <field regex="HOD's Attention" permissions="r" />
        <field regex="Subject" permissions="r" />
        <field regex="Approval of HOD" permissions="rw" />
        <field regex="Allocated Guide" permissions="r" />
</filter-definition>

<filter-definition
        name="readall"
        type="closed"
        add="false"
    >
        <field regex=".*" permissions="r" />
</filter-definition>

</workflow-definition>
```

Listing 4.6 XML Code Listing for regflow__5.6.xml

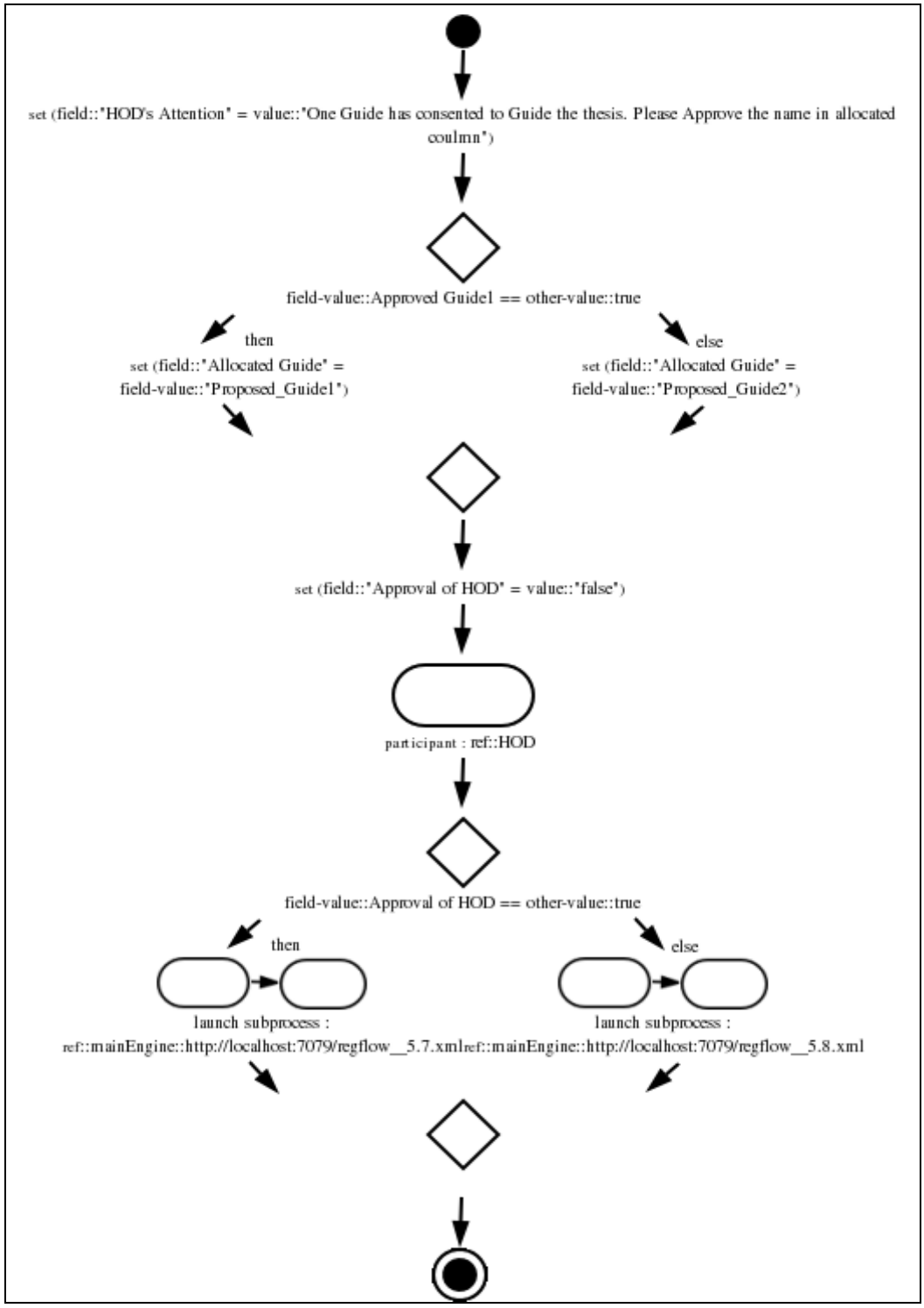

Figure No.4.6 Droflo Equivalent Activity Diagram for regflow__5.6.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<workflow-definition name="regflow" revision="5.7">
<description>This flow informs student about the allotment of Guide</description>

<sequence>
    <set field="Administrator's Attention" value="Guide has been approved by HOD proceed flow to inform student
and guide" />
    <participant ref="Administrator" filter="readall"/>
    <set field-value="Email Id" field="__emailTarget__" />
    <set value="Approval of Thesis Topic" field="__emailSubject__" />
    <set value="Your Thesis Topic: ${field:Thesis Topic}  has been approved. Your Allocated Guide is
${field:Allocated Guide}" field="Students Email Text" />
    <set field-value="Students Email Text" field="__emailText__" />
    <participant ref="email-notif-agent"  />
    <set value="${field:Allocated Guide}@ganit.home.net" field="__emailTarget__" />
    <set value="Guide duties for Thesis" field="__emailSubject__" />
    <set value="You have been assigned as guide for Thesis Titled: ${field:Thesis Topic} for   ${field:Thesis for
BE / ME} as earlier approved by you. The Student details are as follows: Student's Name: ${field:Students Name} ,
Student's Enrolment No:  ${field:Students Enrolment No} ,  Student's Email Id:   ${field:Email Id}" field="Guides
Email Text" />
    <set field-value="Guides Email Text" field="__emailText__" />
    <participant ref="email-notif-agent"  />
    <set value="${field:Students Name}  ${field:Students Enrolment No}" field="__subject__" />
    <participant ref="Archiver"  filter="readall" />
</sequence>

<filter-definition
        name="readall"
        type="closed"
        add="false"
    >
        <field regex=".*" permissions="r" />
</filter-definition>

</workflow-definition>
```

Listing 4.7 XML Code Listing for regflow__5.7.xml

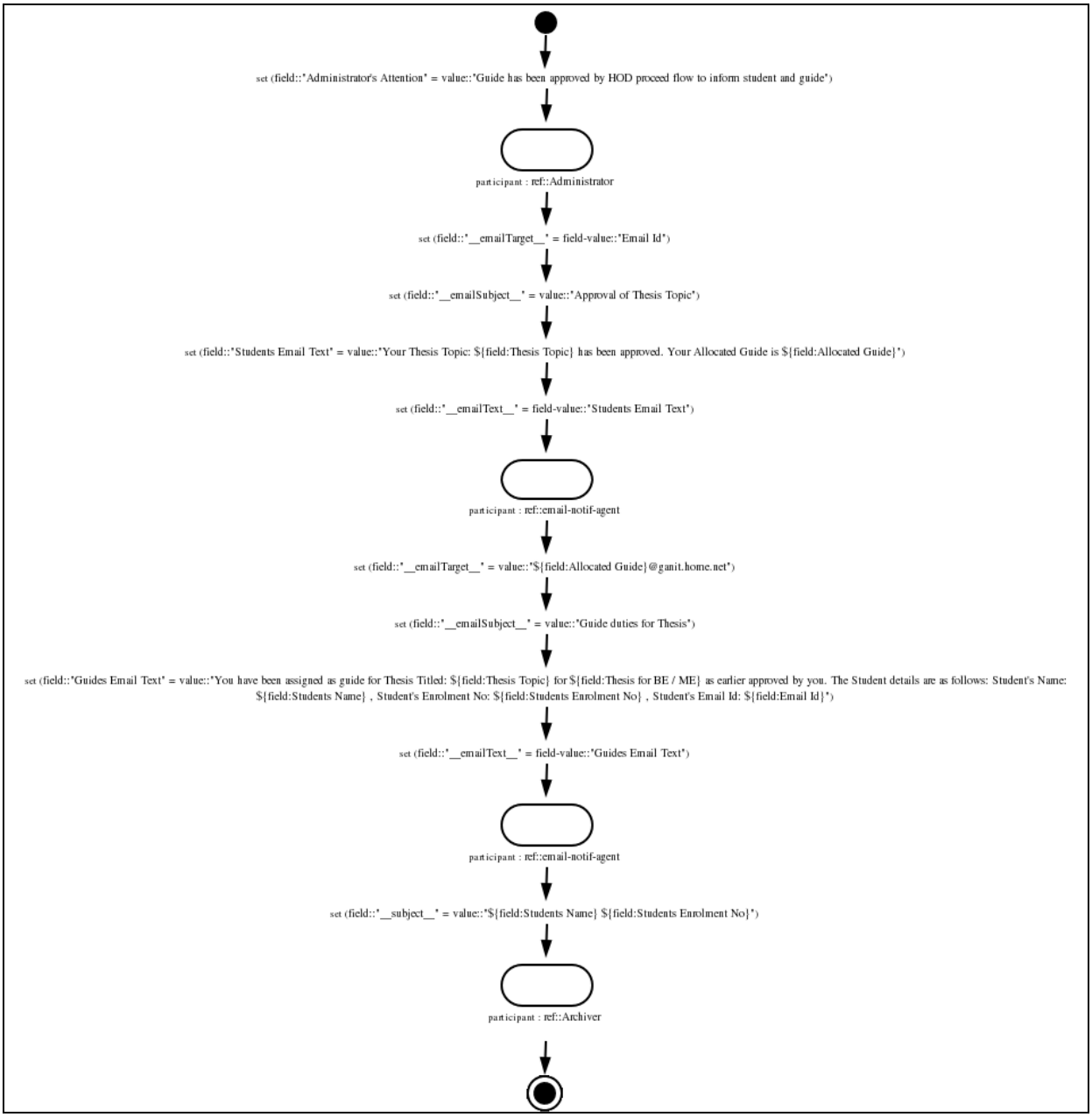

Figure No.4.7 Droflo Equivalent Activity Diagram for regflow__5.7.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<workflow-definition name="regflow" revision="5.8">
<description>This flow sends email to student informing him of non approval</description>
<sequence>
    <set field-value="Email Id" field="__emailTarget__" />
    <set value="Thesis Topic is Not Approved " field="__emailSubject__" />
    <set value="Your Thesis topic has been rejected by the HOD, Please resubmit the topic." field="__emailText__"
/>
    <participant ref="email-notif-agent" description="Thesis topic not accepted" />
    <set value="${field:Students Name}  ${field:Students Enrolment No}" field="__subject__" />
    <participant ref="Thesis" filter="readall"/>
</sequence>
<filter-definition
        name="readall"
        type="closed"
        add="false"
    >
        <field regex=".*" permissions="r" />
</filter-definition>
</workflow-definition>
```

Listing 4.8 XML Code Listing for regflow__5.8.xml

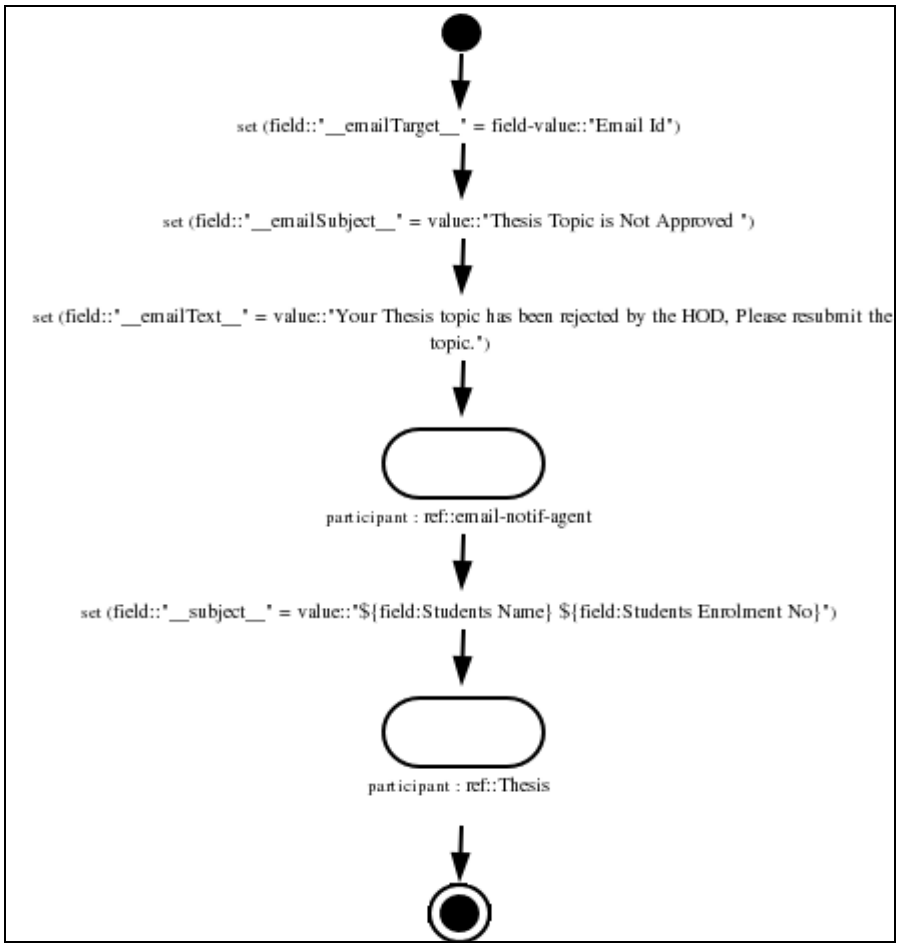

Figure No.4.8 Droflo Equivalent Activity Diagram for regflow__5.8.xml

4.5 Process Implementation Overview

The process implementation overview involves a look at all the components of the workflow system that make the system work as a unit. This involves the flow definition tools, user interfaces, workitem stores, administration and monitoring tools and automated services. OpenWFE uses a flow definition in XML format which have been designed and coded in the work till now. This process definition activity is aided by developing the isssues based on UML tools such as Use Case Diagrams and Activity diagrams. The Activity diagram is then decomposed and coded into XML which the OpenWFE engine understands. Further the OpenWFE requires to know the participants roles that take part in the flow through another XML definition file. It also defines the stores for the work items, users and their access rights using XML definition files. In the subsequent paragraphs each of these implementation aspects will be discussed with the specific problem of thesis registeration as an example.

4.5.1 The Participant Map

OpenWFE separates flow definitions from those who take part in them through the participant map. The engine is interested in participants, it requires a participant map for a dispatcher for the particular participant. The participants are defined in a file called “participantmap.xml” where the participants are linked to a despatcher for that component. The participants can be mapped to a “Role” or a specific “User”. A specific role may have multiple users. In our specific problem the participants are student, professor, administrator, HOD and an email notification agent. The participantmap.xml file is modified with the additional code appended to the file as shown in the code Listing 4.9 below:

```
<participant name="mainEngine">
        <param>
           <param-name>dispatcherClass</param-name>
           <param-value>openwfe.org.engine.impl.dispatch.SocketDispatcher</param-value>
        </param>
        <param>
           <param-name>host</param-name>
           <param-value>127.0.0.1</param-value>
        </param>
```

```
        <param>
          <param-name>port</param-name>
          <param-value>7007</param-value>
        </param>
   </participant>
<participant name="Student">
                  <param>
                    <param-name>dispatcherClass</param-name>
                    <param-value>openwfe.org.engine.impl.dispatch.SocketDispatcher</param-value>
                  </param>
                  <param>
                    <param-name>host</param-name>
                    <param-value>127.0.0.1</param-value>
                  </param>
                  <param>
                    <param-name>port</param-name>
                    <param-value>7008</param-value>
                  </param>
  </participant>
  <participant name="Administrator">
                  <param>
                    <param-name>dispatcherClass</param-name>
                    <param-value>openwfe.org.engine.impl.dispatch.SocketDispatcher</param-value>
                  </param>
                  <param>
                    <param-name>host</param-name>
                    <param-value>127.0.0.1</param-value>
                  </param>
                  <param>
                    <param-name>port</param-name>
                    <param-value>7008</param-value>
                  </param>
  </participant>
  <participant name="Guide1">
                  <param>
                    <param-name>dispatcherClass</param-name>
                    <param-value>openwfe.org.engine.impl.dispatch.SocketDispatcher</param-value>
                  </param>
                  <param>
                    <param-name>host</param-name>
                    <param-value>127.0.0.1</param-value>
                  </param>
                  <param>
                    <param-name>port</param-name>
                    <param-value>7008</param-value>
                  </param>
  </participant>

  <participant name="Guide2">
                  <param>
                    <param-name>dispatcherClass</param-name>
                    <param-value>openwfe.org.engine.impl.dispatch.SocketDispatcher</param-value>
                  </param>
        <param>
```

```
            <param-name>host</param-name>
                  <param-value>127.0.0.1</param-value>
               </param>
               <param>
                  <param-name>port</param-name>
                  <param-value>7008</param-value>
               </param>
</participant>

<participant name="HOD">
               <param>
                  <param-name>dispatcherClass</param-name>
                  <param-value>openwfe.org.engine.impl.dispatch.SocketDispatcher</param-value>
               </param>
               <param>
                  <param-name>host</param-name>
                  <param-value>127.0.0.1</param-value>
               </param>
               <param>
                  <param-name>port</param-name>
                  <param-value>7008</param-value>
               </param>
</participant>

<participant name="Archiver">
               <param>
                  <param-name>dispatcherClass</param-name>
                  <param-value>openwfe.org.engine.impl.dispatch.SocketDispatcher</param-value>
               </param>
               <param>
                  <param-name>host</param-name>
                  <param-value>127.0.0.1</param-value>
               </param>
               <param>
                  <param-name>port</param-name>
                  <param-value>7008</param-value>
               </param>
</participant>
```

Listing 4.9 XML Code Listing for participantmap.xml

The participant role “student” will normally have many users with the user rights for login and launch of registration flow regflow__5.0.xml. Participant role of administrator will have complete rights over the engine, stores and flow launch. Participant roles Guide1 and Guide2 will normally be mapped to individual professor and may be annotated by professor’s name as the participant name. In context of problem under design we have used generic nomenclature. HOD participant role can be attributed to any specific professor by granting rights to the HOD store.

4.5.2 The Stores

The stores are defined places where workitems are placed. Each store is a service and has attributes like lockout time and unlock frequency associated with it. In the problem under consideration the stores as given in Listing 4.10 are defined and appended to a file called worklist-configuration.xml.

```
<service name="Store.Thesis"
        class="openwfe.org.worklist.impl.store.SimpleWorkItemStore">
    <!--
          workitems if they are not approved are put in this worklist administrator will have access to this
        -->
                <param>
                  <param-name>participants</param-name>
                  <param-value>Thesis</param-value>
                </param>
                <param>
                  <param-name>lockTimeout</param-name>
                  <param-value>15m</param-value>
                </param>
                <param>
                  <param-name>unlockFrequency</param-name>
                  <param-value>3m</param-value>
                </param>
 </service>

 <service name="Store.Administrator"
        class="openwfe.org.worklist.impl.store.SimpleWorkItemStore">
    <!--
          workitems for participant 'administrator' are put in this worklist
        -->
                <param>
                  <param-name>participants</param-name>
                  <param-value>Administrator</param-value>
                </param>
                <param>
                  <param-name>lockTimeout</param-name>
                  <param-value>15m</param-value>
                </param>
                <param>
                  <param-name>unlockFrequency</param-name>
                  <param-value>3m</param-value>
                </param>
 </service>
 <service
    name="Store.Guide1"
    class="openwfe.org.worklist.impl.store.SimpleWorkItemStore"
 >
    <!--
       workitems for participant 'professor' are put in this worklist
```

```
      -->
      <param>
        <param-name>participants</param-name>
        <param-value>Guide1</param-value>
      </param>
      <param>
        <param-name>lockTimeout</param-name>
        <param-value>15m</param-value>
      </param>
      <param>
        <param-name>unlockFrequency</param-name>
        <param-value>3m</param-value>
      </param>
    </service>

    <service
      name="Store.Guide2"
      class="openwfe.org.worklist.impl.store.SimpleWorkItemStore"
    >
      <!--
        workitems for participant 'Guide2' are put in this worklist
      -->
      <param>
        <param-name>participants</param-name>
        <param-value>Guide2</param-value>
      </param>
      <param>
        <param-name>lockTimeout</param-name>
        <param-value>15m</param-value>
      </param>
      <param>
        <param-name>unlockFrequency</param-name>
        <param-value>3m</param-value>
      </param>
    </service>
<service
      name="Store.HOD"
      class="openwfe.org.worklist.impl.store.SimpleWorkItemStore"
    >
      <!--
        workitems for participant 'hod' are put in this worklist
      -->
      <param>
        <param-name>participants</param-name>
        <param-value>HOD</param-value>
      </param>
      <param>
        <param-name>lockTimeout</param-name>
        <param-value>15m</param-value>
      </param>
      <param>
        <param-name>unlockFrequency</param-name>
        <param-value>3m</param-value>
      </param>
    </service>
```

```
<service
    name="Store.Archiver"
    class="openwfe.org.worklist.impl.store.SimpleWorkItemStore"
 >
    <!--
       workitems that are approved are put in this worklist
    -->
    <param>
       <param-name>participants</param-name>
       <param-value>Archiver</param-value>
    </param>
    <param>
       <param-name>lockTimeout</param-name>
       <param-value>15m</param-value>
    </param>
    <param>
       <param-name>unlockFrequency</param-name>
       <param-value>3m</param-value>
    </param>
 </service>
```

Listing 4.10 XML Code Listing for worklist-configuration.xml

4.5.3 The Users and Rights

The stores and flow definitions are the two critical aspects of the workflow engine that need users to inteact to execute useful work. First of all the users need to be defined for the system and then each user is to be given access to view a workitem store that contains work for him/her. Further a few users will need the rights to launch a flow as in our case the student launches the flow for registration of thesis topic. The users and rights are defined in a file called passwd.xml and in our case the code for file is given in Listing 4.11 below:

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
   $Id: passwd.xml,v 1.5 2005/07/03 12:31:03 Simmi Dutta Exp $

   You can generate the password hashes with ./genpass.sh or genpass.bat
-->

<passwd>

  <principal
          name="simmi"
          class="openwfe.org.auth.BasicPrincipal"
```

```
        password="-87-118+17+81-89+0-74-2-85-91-49-1-12-107+1-50"

>
        <grant name="launch.regflow" />

</principal>

        <principal
        name="rajeev"
        class="openwfe.org.auth.BasicPrincipal"
        password="-90+117-1-78-20+36-109+111-70+110+22-43-13-124+87-113"

>
        <grant name="launch.regflow" />

</principal>

<principal
        name="goldie"
        class="openwfe.org.auth.BasicPrincipal"
        password="-58-46+82+117-77+123+33+91+0+93+126-8+39+61+111-96"

>

        <grant name="store.Guide1" />
</principal>

<principal
        name="sks"
        class="openwfe.org.auth.BasicPrincipal"
        password="-51+87+6-83+21-71+95-105-11-108+52-1-114-45-55+118"

>

        <grant name="store.Guide2" />
</principal>

<principal
        name="drc"
        class="openwfe.org.auth.BasicPrincipal"
        password="-88-13+61-21+108+65-54+39-66+118-7-62+100-107+21-106"

>
                <grant name="store.HOD" />
</principal>

<principal
        name="admin1"
        class="openwfe.org.auth.BasicPrincipal"
        password="-32+12-14+90-44+38-125-77-33+103-116+97-12+44+107-38"

>
        <grant name="store.Guide1" />
        <grant name="store.Administrator" />
        <grant name="store.Guide2" />
```

```
   <grant name="store.HOD" />
        <grant name="store.Archiver" />
        <grant name="store.Thesis" />
        <grant name="launch.regflow" />
</principal>

<!-- grants -->

<!--
        Grants are sets of permissions.
        They apply to a codebase (a jar file) a set of permissions.

        (Worklist-wide permissions are granted in
        etc/worklist/worklist-policy.conf or
        etc/worklist/win-worklist-policy.conf (for windows)

        The 'name' of a permission is usually the name of the object
        to which it grants some kind of access right.
        For a StorePermission, the name is the name of the store concerned.
-->

<grant name="store.Administrator"
        codebase="file:./jars/openwfe-worklist-actions.jar"
>
        <permission
                name="Store.Administrator"
                class="openwfe.org.worklist.auth.StorePermission"
                rights="read, write, delegate"
        />
</grant>
<grant name="store.Guide1"
   codebase="file:./jars/openwfe-worklist-actions.jar"
>
  <permission
     name="Store.Guide1"
     class="openwfe.org.worklist.auth.StorePermission"
     rights="read, write, delegate"
   />
</grant>
        <grant name="store.Guide2"
   codebase="file:./jars/openwfe-worklist-actions.jar"
>
  <permission
     name="Store.Guide2"
     class="openwfe.org.worklist.auth.StorePermission"
     rights="read, write, delegate"
   />
</grant>
<grant name="store.HOD"
   codebase="file:./jars/openwfe-worklist-actions.jar"
>
  <permission
```

```
        name="Store.HOD"
         class="openwfe.org.worklist.auth.StorePermission"
         rights="read, write, delegate"
      />
   </grant>
   <grant name="store.Archiver"
      codebase="file:./jars/openwfe-worklist-actions.jar"
   >
     <permission
        name="Store.Archiver"
        class="openwfe.org.worklist.auth.StorePermission"
        rights="read, write, delegate"
      />
   </grant>
   <grant name="store.Thesis"
      codebase="file:./jars/openwfe-worklist-actions.jar"
   >
     <permission
        name="Store.Thesis"
        class="openwfe.org.worklist.auth.StorePermission"
        rights="read, write, delegate"
      />
   </grant>
   <grant name="launch.regflow"
                                codebase="file:./jars/openwfe-worklist-actions.jar"
   >
          <permission
             name="mainEngine::http://localhost:7079/regflow__5.1.xml"
             class="openwfe.org.worklist.auth.LaunchPermission"
           />
          <permission
             name="mainEngine::http://localhost:7079/regflow__5.0.xml"
             class="openwfe.org.worklist.auth.LaunchPermission"
          />
   </grant>
</passwd>
```

Listing 4.11 XML Code Listing for passwd.xml

It can be clearly seen in the listing above that the launch and store permissions are given to each user defined. This is generally done at the worklist level and a facility exists to control this process graphically using a web interface.

4.5.4 Ancillary Services Implementation

The demonstration prototype required a host of support services for sucessfully implementing a workflow management system. This involved an operating environment that was multiuser

capable, a webserver, a mailserver and a java virtual machine. Fedora Core 3 operating system was configured as the basic OS because of its open source availability. Apache webserver was setup on this linux system with the host name as ganit.home.net and a class C IP address of 192.168.0.1. A mail server using sendmail and SMTP was setup as the core mail processing utility. The mail was to be available through web iterface, so a webmail utility using Squirell mail was also setup.

4.6 WFMS Results

The workflow system was implemented and configured based on the discussions in the previous paragraphs. The flow definitions were implemented in XML as presented after placing them appropriately in the software file hierarchy. The configuration files were modified as stated earlier to implement a solution for our problem.

4.6.1 Apache Webserver Homepage

The first login was provided a hook from the local webserver root page as seen in figure 4.9. The system branched to the worklist handler that was hyperlinked to the workflow management services caption on the main page and landed at the concerned handler port http://192.168.0.1:7080 at the weclient service.

4.6.2 WebClient Interface

This service was the frontend for all the users of the WFMS. The interface picture is shown in Figure No.4.10. Any user requiring a login into the WFMS would require to enter the system through this interface.

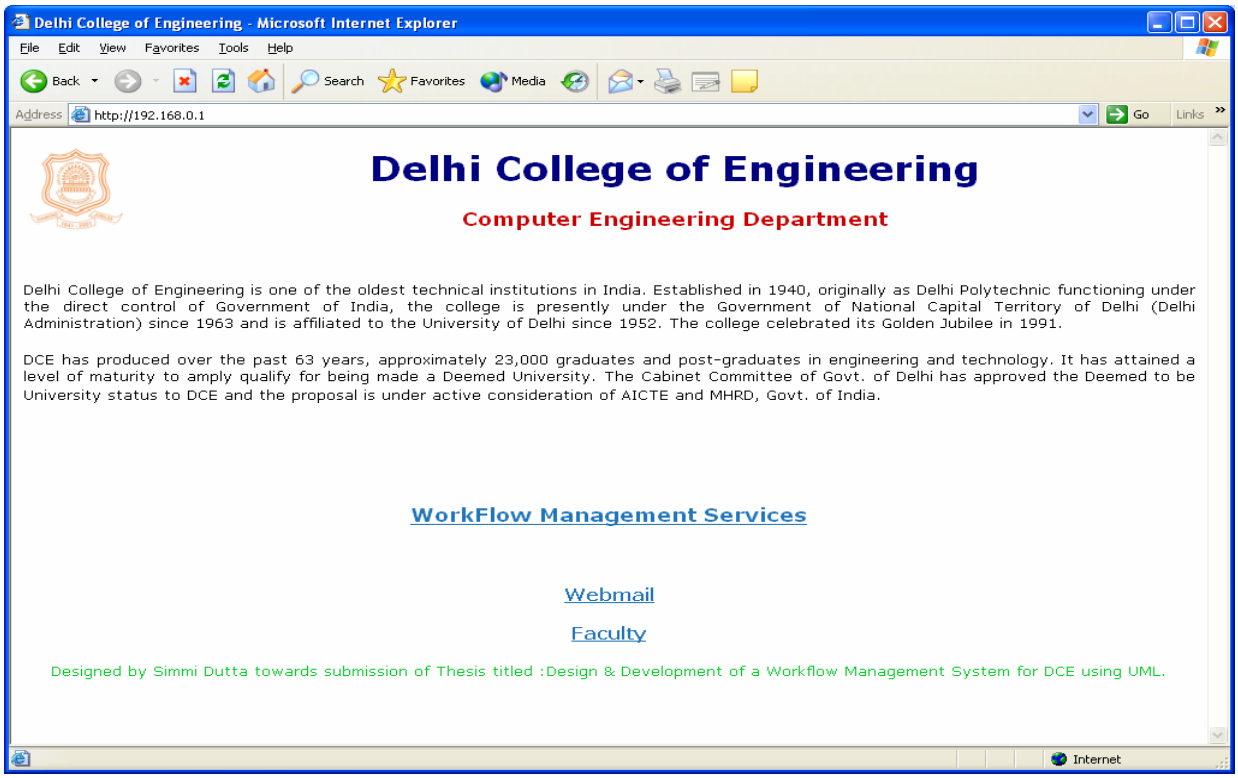

Figure No.4.9 Main Webpage providing link to Workflow Management

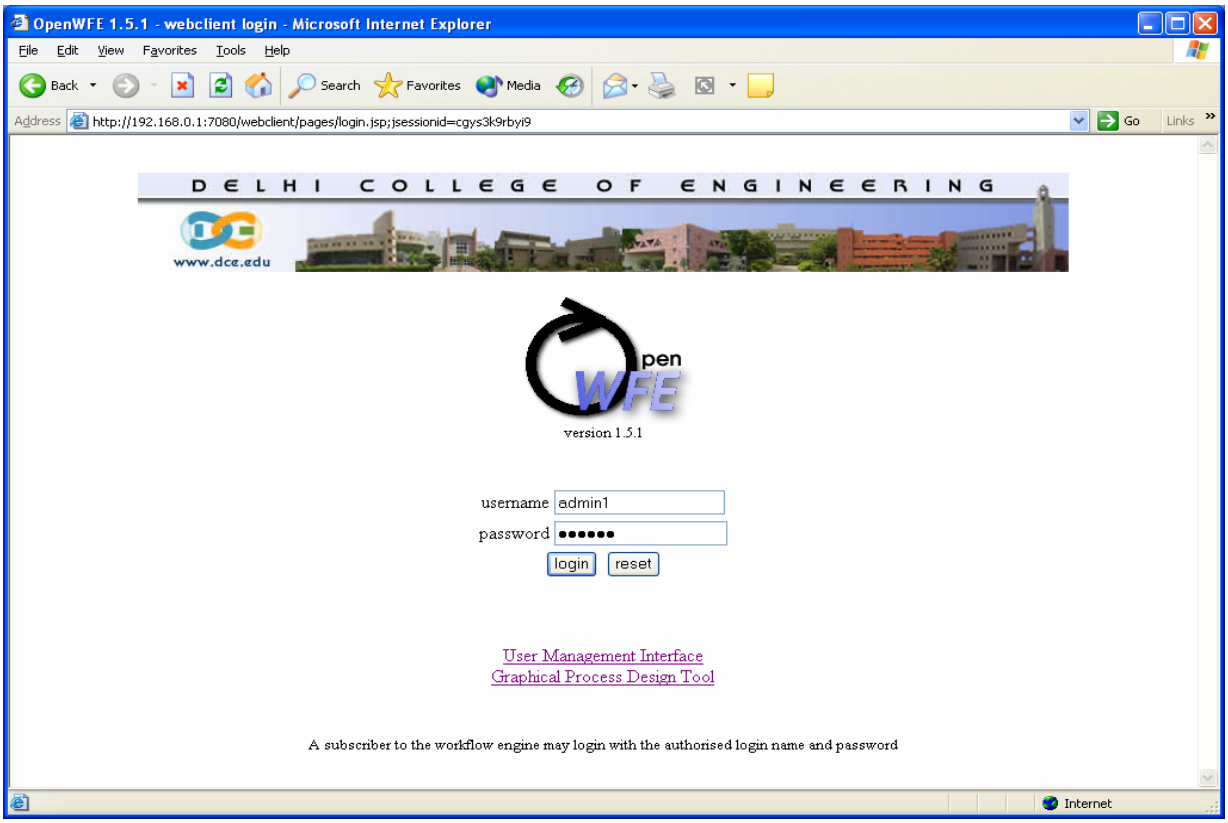

Figure No.4.10 Webclient Interface for Login into WFMS

The webclient used in the WFMS is a standard webbrowser that can access the site over a lan or internet at a designated IP addresss . The client browser could be Internet Explorer or Mozilla or Opera and it uses a HTTP request to interact with the WFMS. The webclient interface is generated using a JSP script and hyper-links User Management Interface and process design helper utility.

4.6.3 Stores Information and Flow Launch Page

After an authenticated login, the user would landup at a page where the stores status is depicted. Only the store to which he has rights is shown with the corresponding workitems in it. The user has an option to view or edit the workitem. The user can launch the flow definitions by going to a flow list page by using the hyperlink provided on this page. Figure 4.11 shows the stores page as will be seen by an administrator. The tables for each store are empty indicating no pending workitems except store of administrator where one pending workitem is shown with the controlling flow nomenclature and despatch time.

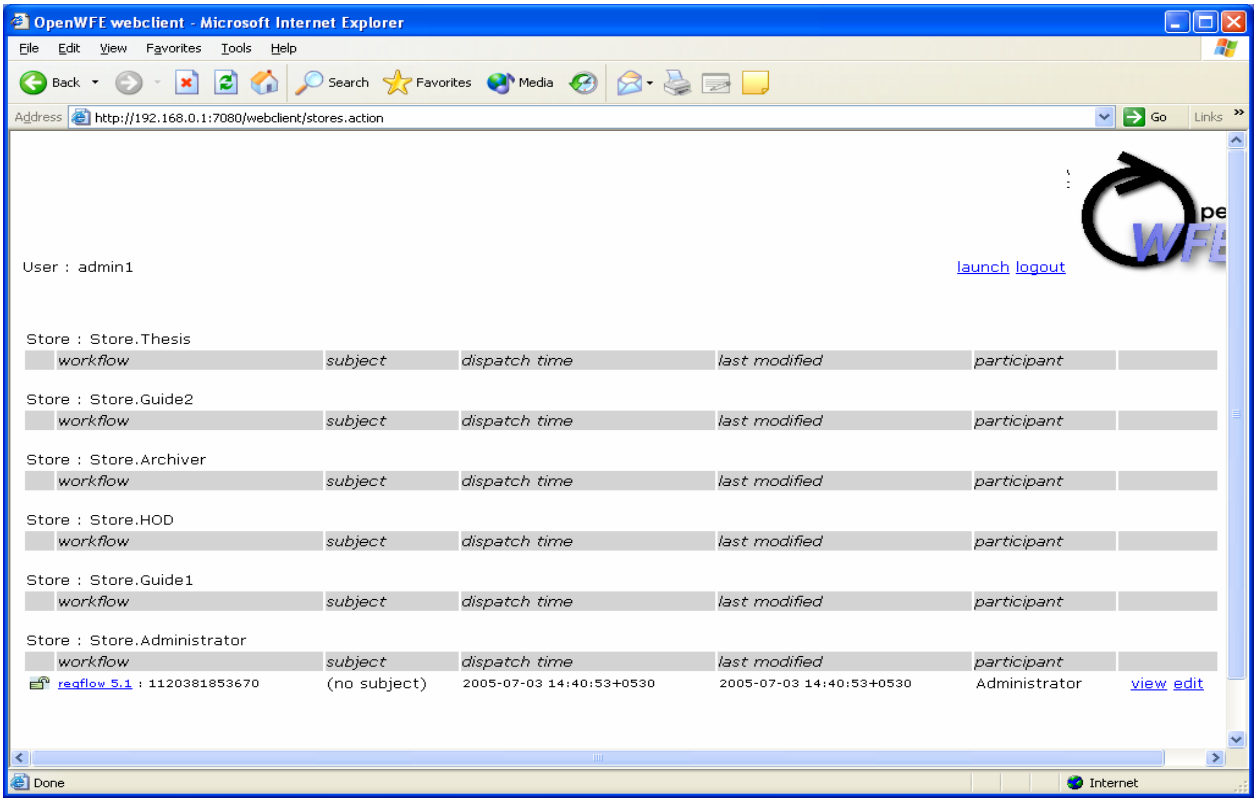

Figure No.4.11 Store Interface for Administrator

The launch page as seen by a student is shown in Figure No. 4.12. This page will list the flow definitions and subjects that are permitted to be launched by a student. This page shows the launch permissions for a user named *simmi*.

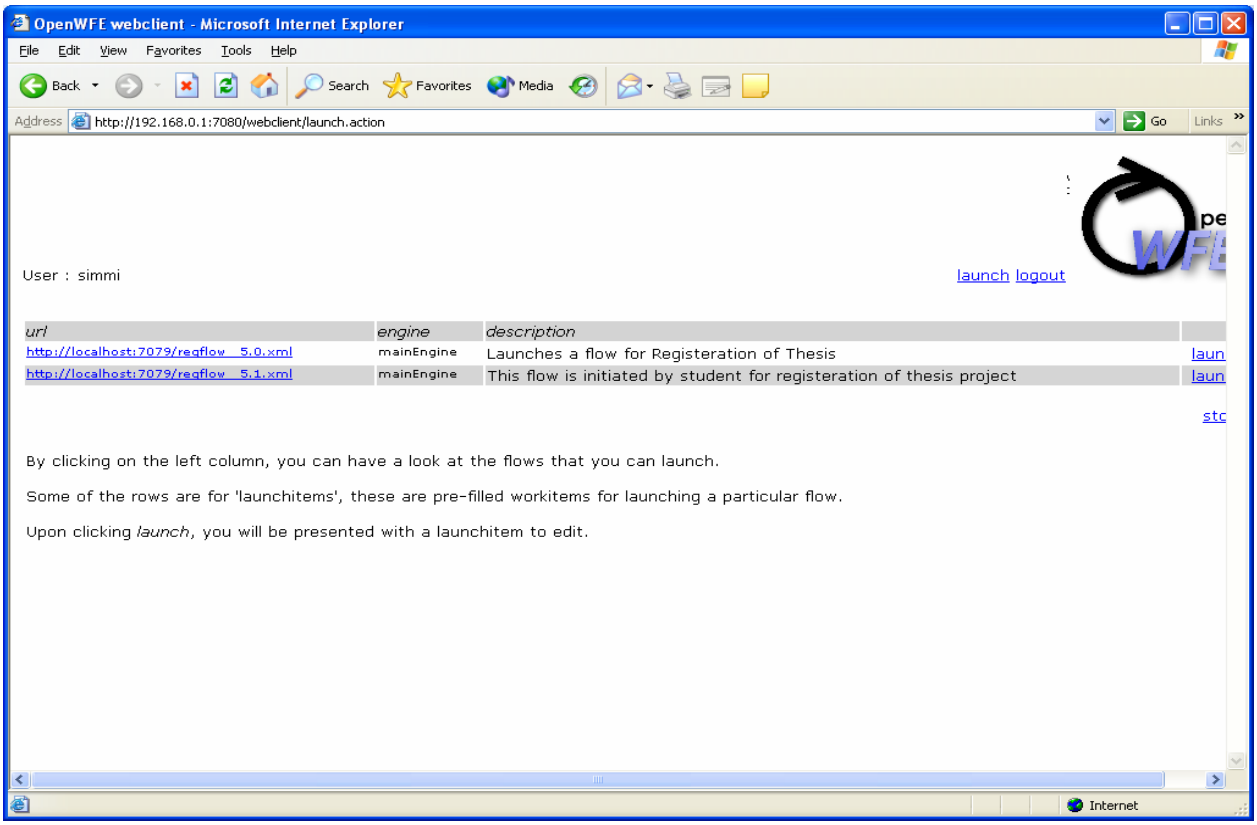

Figure No.4.12 Flow Launch Interface

4.6.4 Workitem Edit Interface

Once the flow has been launched, the primary workitem is generated and it is presented in an edit mode to the user launching the flow. The user can fill in certain design time parameters and launch the flow to the next step in the process. A diagram of workitem presented to student user at launch of flow “regflow__5.0.xml” is shown in Figure No.4.13.

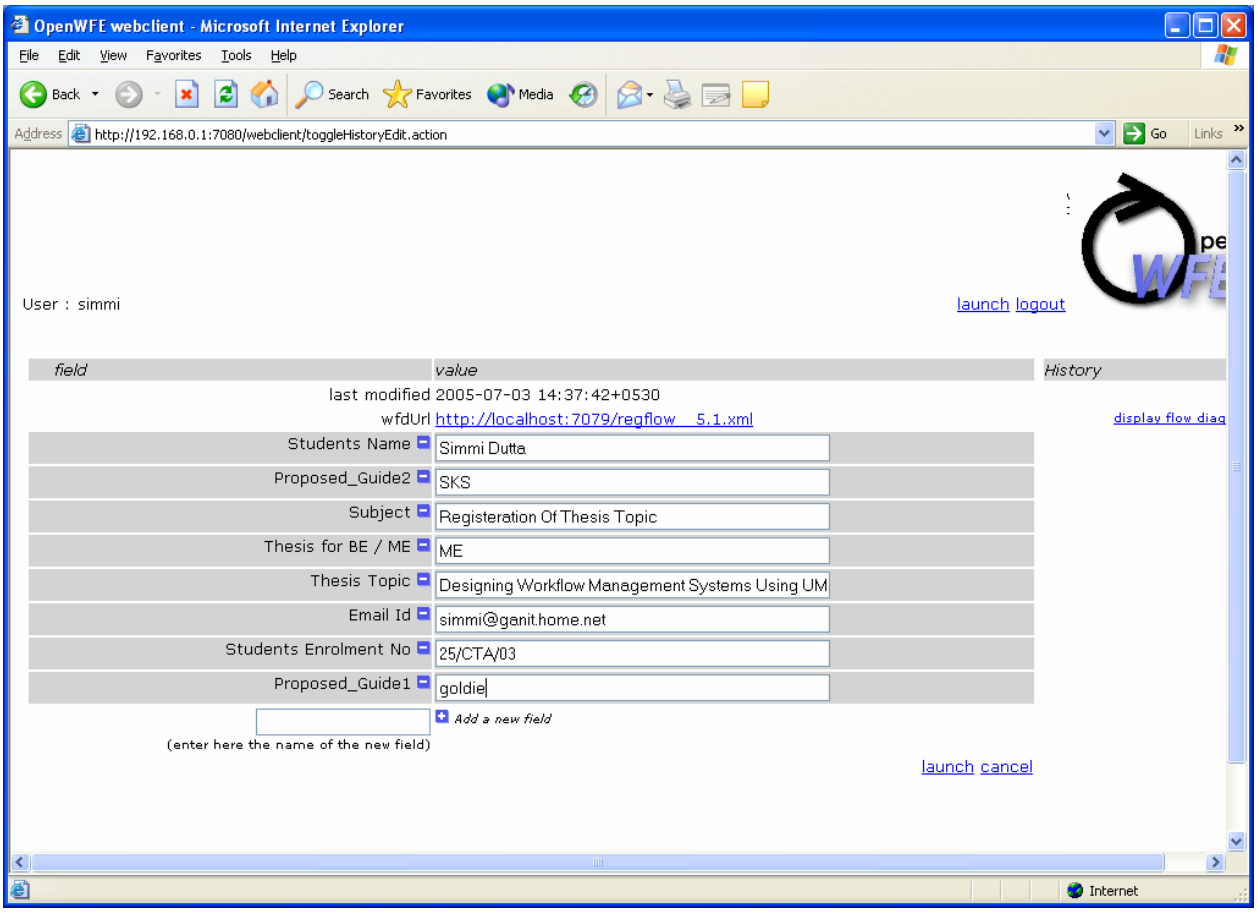

Figure No.4.13 Workitem Launch Time Interface

Once the initial details have been filled up the user will launch the workitem and it will show up in the store of the next participant in the flow. In this case the next participant is administrator and the store will be as seen in Figure No. 4.11. The administrators role at this step is to either approve or disapprove the thesis topic. The flow will take relevant path based on this decision as per Figure No.4.1. To endorse administrators action the administartor needs to edit the work item. The administrators edit interface will be as shown in Figure No.4.14 and he would make a boolean selection on the only editable field presented to him for approval.

4.6.5 The Process Flow

On condition of non approval being met an email is sent through the email-notify-agent and flow regflow__5.3.xml as shown in Figure No.4.3. The email massage received by the student is

shown in his webmail box as presented in Figure No. 4.15 and subsequently the rejected topic workitem is deposited in store Thesis for administrators persual. If the approval is granted, flow goes through to the process regflow__5.2.xml. The workitem is now split into two and routed parallelly to the two professors and each professor gets a workitem in their respective allocated Guide store.

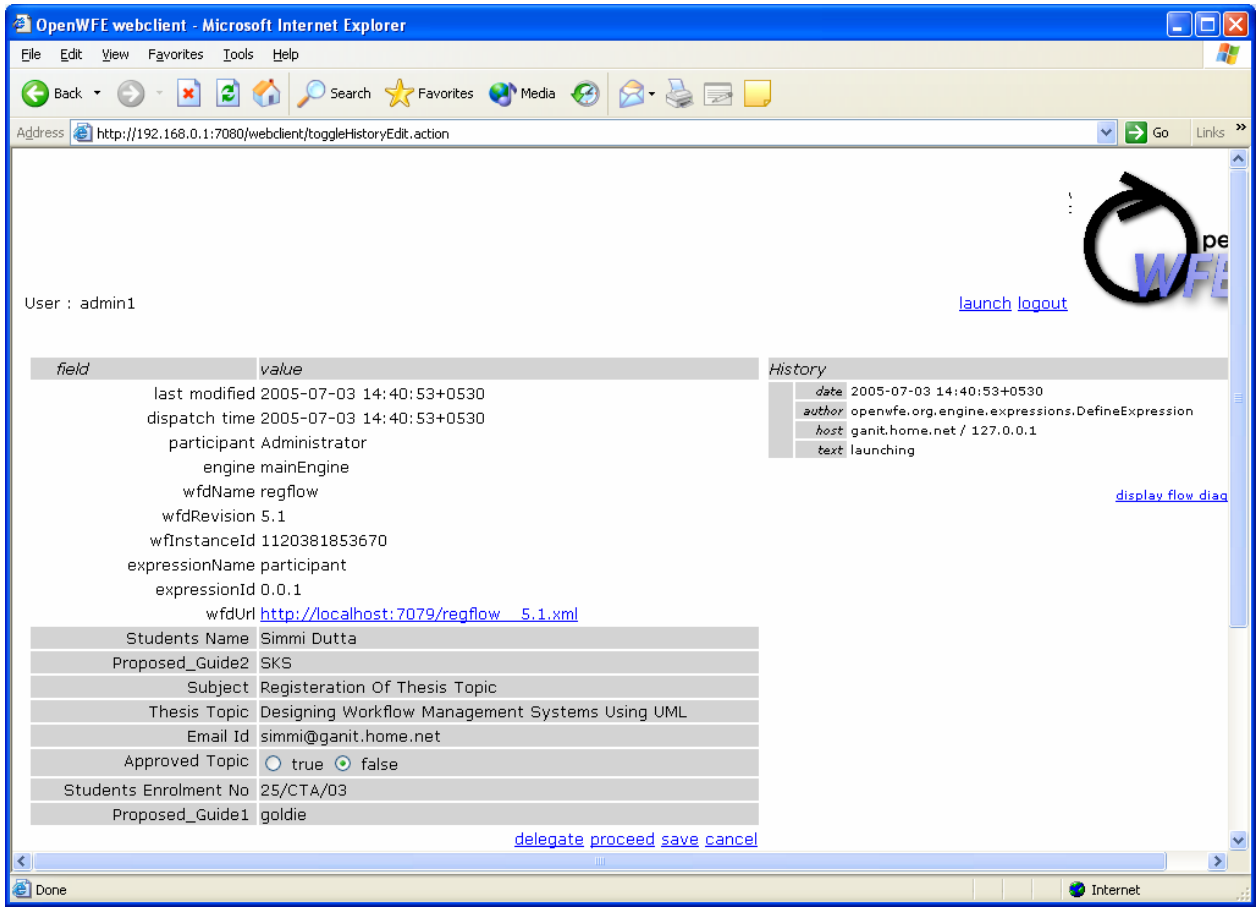

Figure No.4.14 Administrator's Workitem Edit Interface

The workitem received in each guide store is edited by the professor, by accepting to guide the work or not. The guide store and edited workitem is shown in figure 4.16 and figure 4.17 respectively.

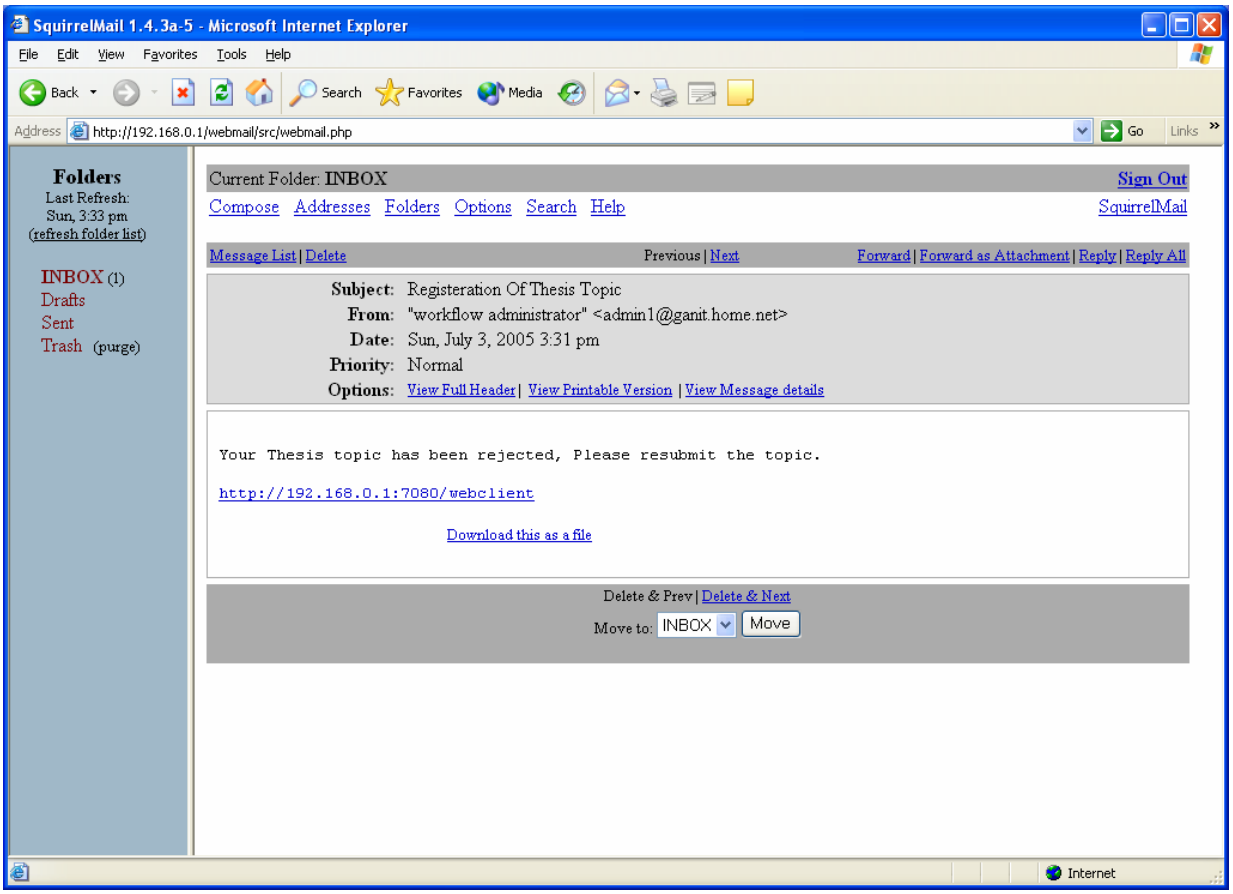

Figure No.4.15 Students Webmail Inbox

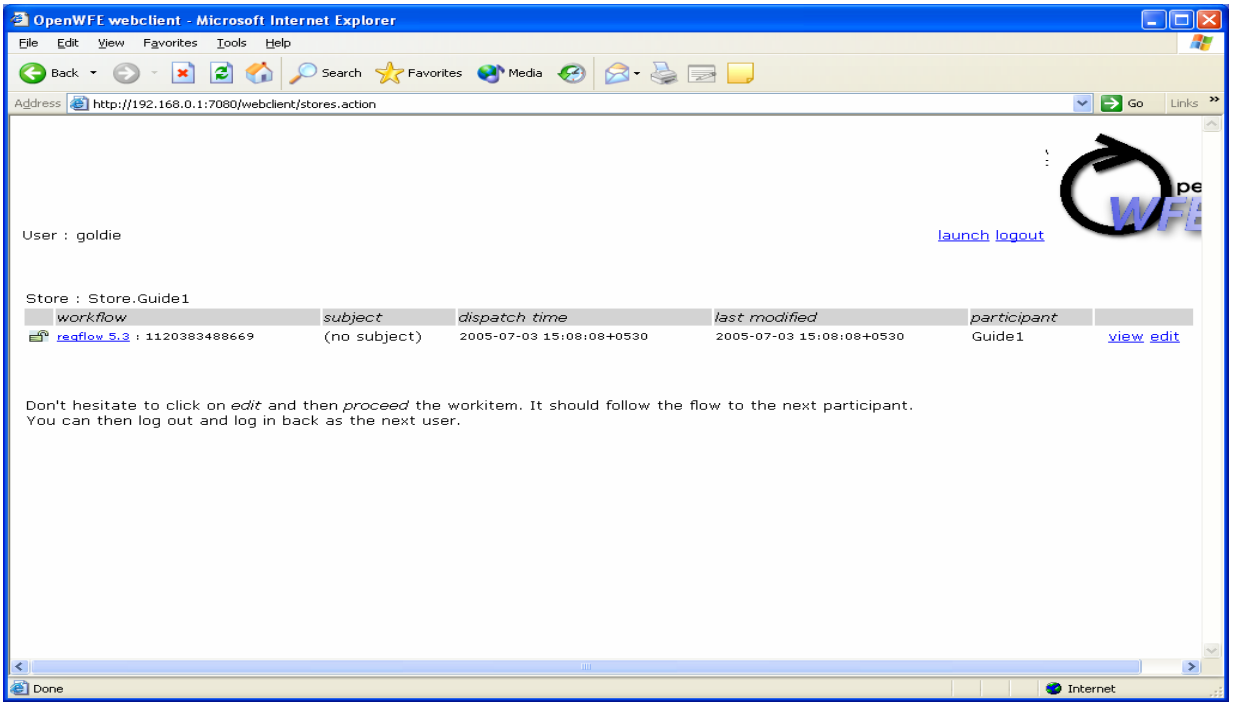

Figure No.4.16 Guide Store

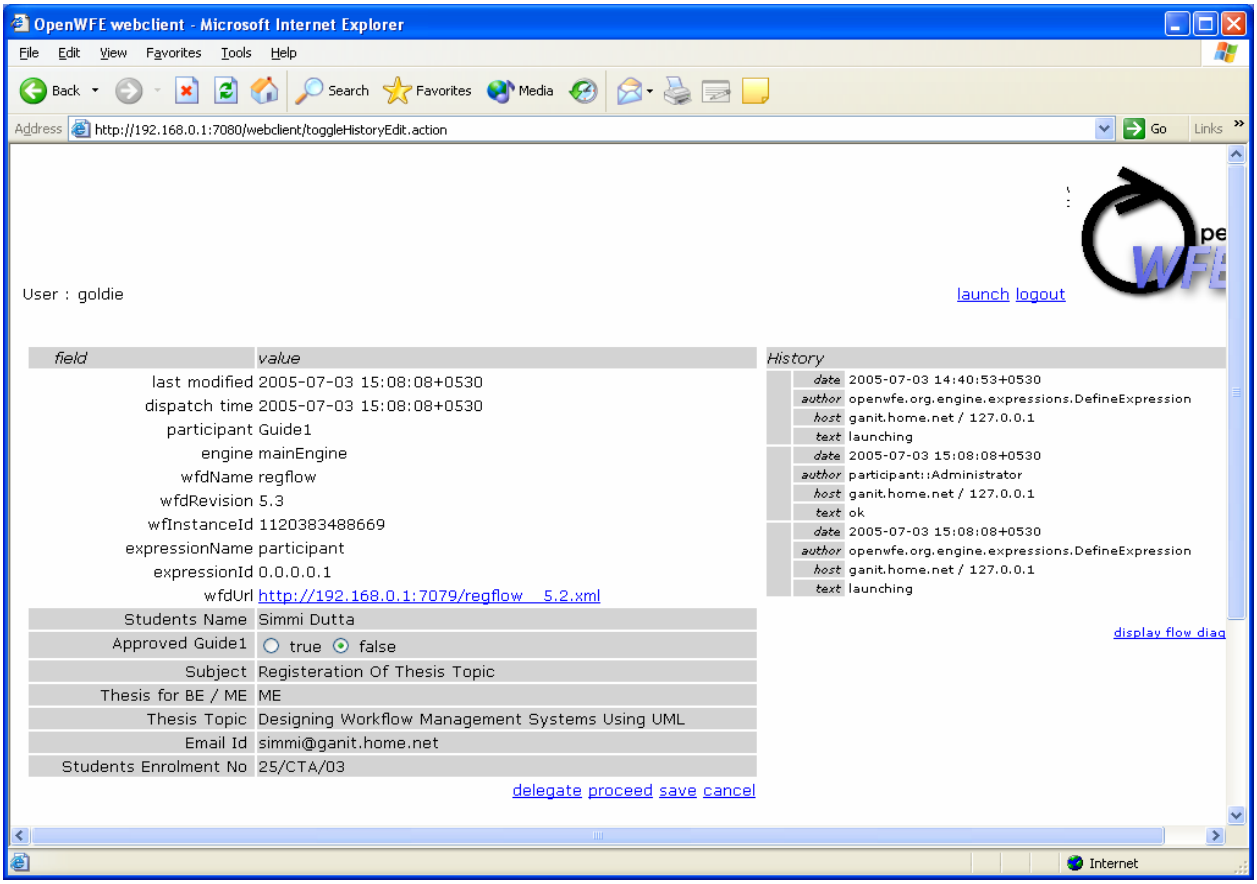

Figure No.4.17 Guide Workitem Edit Menu

Once the flow has gone across both guides, if arbitration is required it goes through the subprocess regflow__5.4.xml where the HOD allocates a guide, else HOD approves the guide who has consented. The form presented to HOD for arbitration is shown in Figure No.4.18. Activity diagrams in Figure Nos. 4.4, 4.5 or 4.6 are followed depending on the approval of guides. Once the approval of HOD is acquired the workitem is progressed to the administrator and an email is sent to student and the approved guide accordingly. The webmail inbox of student and guide are shown in Figure Nos. 4.19 and 4.20 respectively. Finally the workitem is sent to achiever for storage and further reference by administrator.

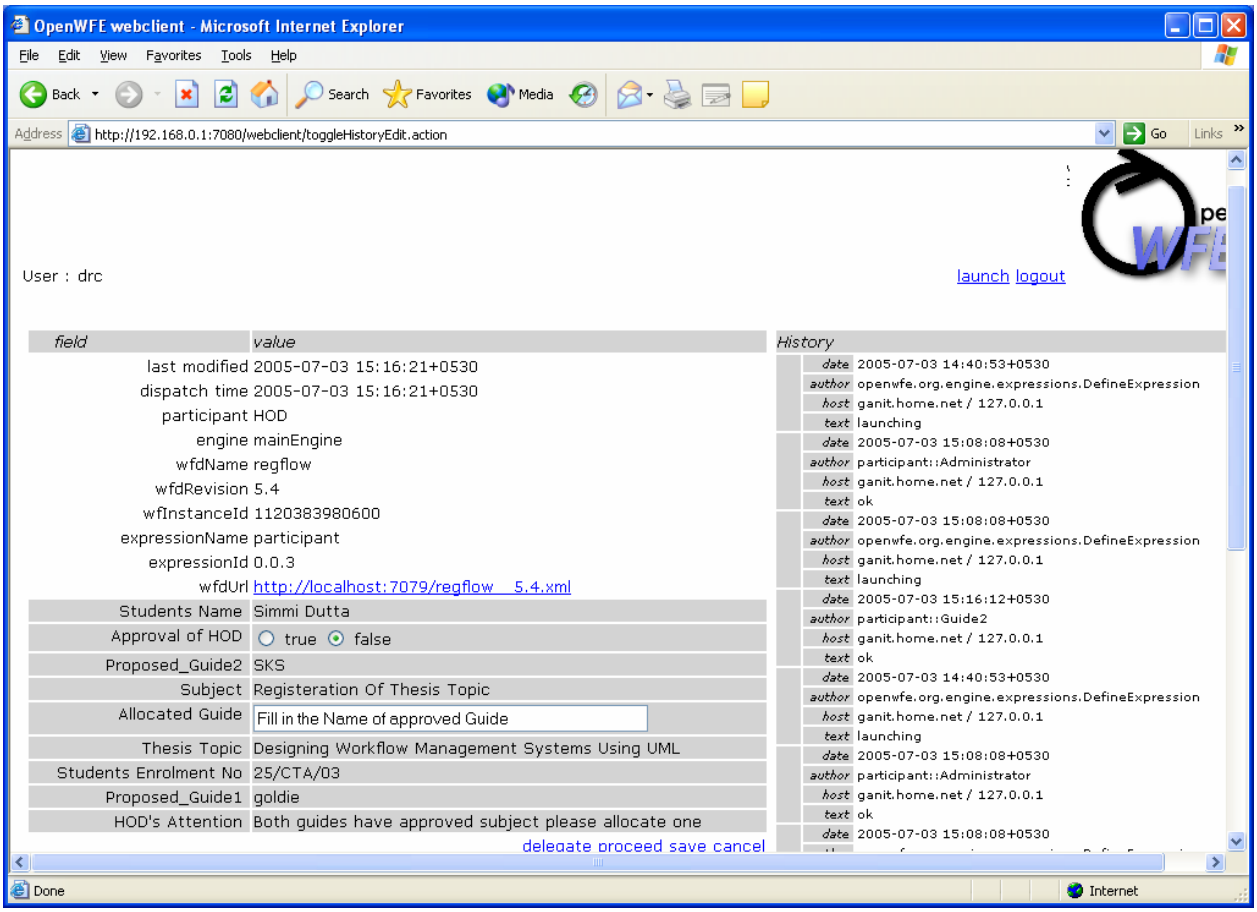

Figure No. 4.18 HOD Workitem Edit Form

If for any reason both the professors refuse to guide the thesis work, or the HOD does not approve the thesis topic or guides, an email is sent to the student about the rejection and the disapproved thesis workitem is sent to thesis store. At the end of the flow if the thesis topic is approved and the guide allocated, the workitem will reside in the Archiver store else it would be sent to the Thesis store. The email is sent to student in case of rejection of thesis topic by both guides or HOD denying approval.

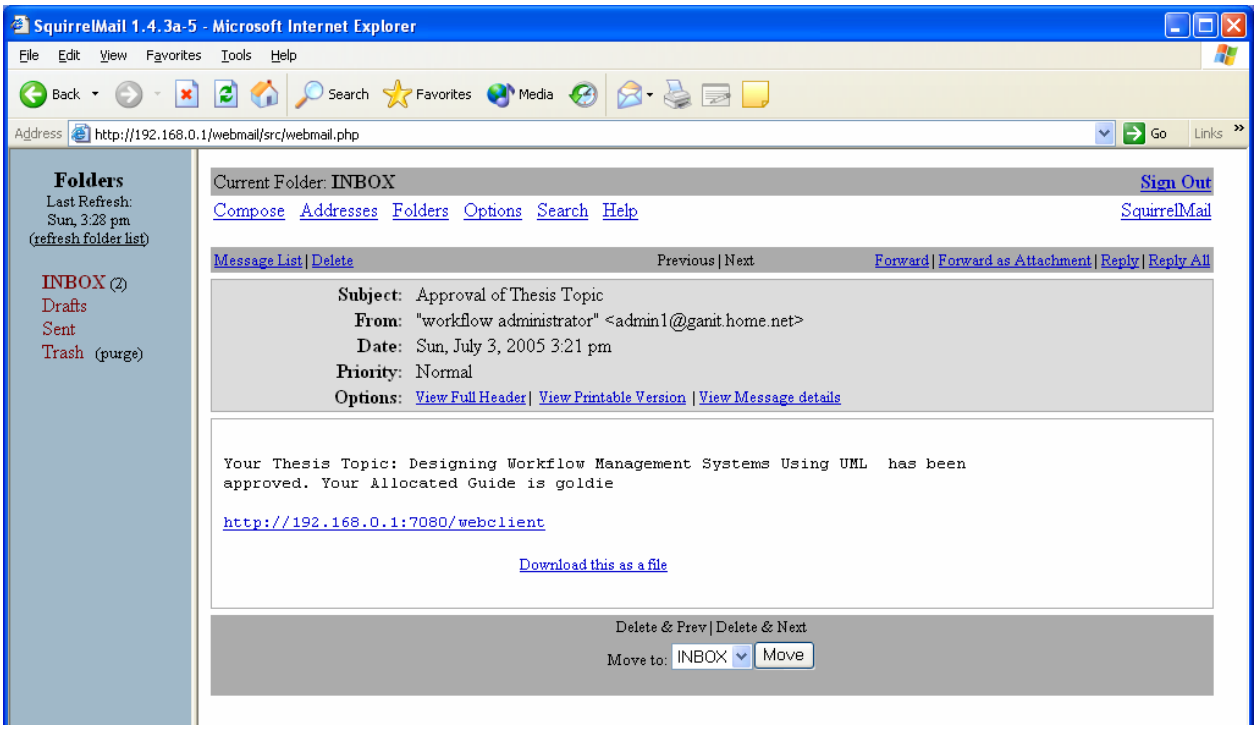

Figure No. 4.19 Webmail to student on Approval

Figure No. 4.20 Webmail to Allocated Guide on Approval

Chapter 5

Conclusions and Future Scope

5.1 Conclusions

The aim of this work was to design and develop a Workflow Management System for Delhi College of Engineering using Unified Modeling Language. Within the scope of this work UML techniques involving Use Case analysis and Activity Diagrams were applied to analyse and design the Workflow in the College Computer Engineering Department. This preliminary design formed the basis of deployment of organizational process in workflow centric approach on a Workflow Management System.

Detailed infrastructure for the WFMS was integrated from a variety of software available in the open source category so as to minimize cost of ownership of such a WFMS. Guidelines laid down by WFMC (Workflow Management Coalition) were used as standardization for the work. The complete infrastructure was implemented on top of a Linux Box running Fedora Core 3 using OpenWFE as open source workflow engine.

The Organizational process of a student registering for thesis was taken as a sample process for design and implementation. This process was implemented successfully using XML code driven from the UML activity diagrams for the process. A working prototype system was implemented and demonstrated.

Within the scope of this work a detailed infrastructure for implementing any workflow process has been evolved, tested and prototyped. Further organizational processes can be easily added to the WFMS using the prototyped infrastructure.

5.2 Future Scope

The Workflow management strategy presented here is an open source, low cost, net enabled implementation. This in comparison to the existing WFMS is flexible and scaleable. The infrastructure can be used for further future developments and utilisation as follows:

(f) Design and developemnt of additional processes within Delhi College of Engineering.

(g) Design and Development of additional Python based APRE agents for executing additional system controlled automated tasks.

(h) Design and development of a relational database link for management of large volumes of workflow data and users.

(i) Advancing the WFMS for embedding into current groupware applications as well as relational databases.

REFERENCES

[1]. Introduction to Workflow Charles Plesums Computer Sciences Corporation, Financial Services Group

[2]. Rob Allen, Open Image Systems Inc., United Kingdom Chair, WFMC External Relations Committee, Workflow: An Introduction.

[3]. Clarence A. Ellis and Gary J. Nutt. Modeling and Enactment of Workflow Systems, pages 1–16.Invited paper.

[4]. C. Mohan. Recent trends in workflow management products, standards and research, pages 396–409.Volume 164 of Do¢ gaç et al. [27], August 1998.

[5]. David Hollingsworth, Technical Committee, WFMC, the Workflow Reference Model 10 Years On.

[6]. David Hollingsworth. The Workflow Reference Model. Workflow Management Coalition, Avenue Marcel Thiry 204, 1200 Brussels, Belgium, 1995.

[7]. Amit P. Sheth, Wil van der Aalst, and Ismailcem B. Arpinar. Processes driving the networked economy. IEEE Concurrency, pages 18–31, July–September 1999.

[8]. Dimitrios Georgakopoulos, Mark Hornick, and Amit Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. Distributed and Parallel Databases, an International Journal, 3:119–153, 1995.

[9]. Hewlett-Packard. HP Changengine–Business Process Management for the Enterprise. Available on the Web at http://www.ice.hp.com/cyc/af/00/index.html/

[10]. Meteor (UGA/LSDIS), http://lsdis.cs.uga.edu.proj/meteor/meteor.html/

[11]. Yanbo Han, Amit Sheth, and Christoph Bussler. A taxonomy of adaptive workflow management.CSCW Towards Adaptive Workflow Systems Workshop, Seattle,WA, November 1998.

[12]. WFMC, “Interface1: Process Definition Model and Interchange Language V 1.1 Final (WfMC-TC-1016-P)," October 1999, http://www.wfmc.org/

[13]. Details at www.omg.org/

[14]. Santanu Paul, Edwin Park, and Jarir Chaar. Essential requirements for a workflow standard.OOP-SLA’97 Business Object Workshop, 1997.

[15]. e-workflow – the workflow portal. A dedicated workflow website sponsored by the WFMC. This site has links to many other workflow related sites. http://www.e-workflow.org

[16]. Oracle Workflow. Visit the Oracle web site and look for information on Oracle Workflow. http://www.oracle.com/

[17].http://otn.oracle.com/products/ias/workflow/workflow_fov.html -Workflow Feature Overview

[18]. BEA WebLogic Integration. Visit the BEA website and look for information on WebLogic Integration. http://www.bea.com

[19]. JBoss jBPM. A recent addition to the JBoss family of middleware products to Add business process management. http://www.jbpm.org/

[20]. Enhydra Shark. An open source workflow engine. A Java/XML based workflow engine based on WfMC standards and XPDL. http://sharkobjectweb.org/

[21]. OpenSymphony OSWorkflow. A low-level, open source, workflow system. http://www.opensymphony.com/

[22]. OpenWF. A third party workflow engine and graphical design package based onMicrosoft .Net technology http://www.openWF.com/

[23]. Jay Lawrence, OpenWFE WorkFlow Developers Guide, v.0.0- Sep, 2004

[24]. Marlon Dumas & Arthur H.M. ter Hofstede: UML Activity Diagrams as a WorkFlow specification Language.

[25]. W.M.P. Van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski and A.P. Barros: WorkFlow Patterns.

[26]. Workflow Patterns. An academic web site dedicated to the study of workflow modelling patterns http://tmitwww.tm.tue.nl/research/patterns/index.htm

BOOKS REFERENCED

1 John Mettraux, OpenWFE, Open Source WorkFlow Engine .
Available on the web at http://www.openwfe-1.5.1/doc/book_openwfe.html/

2 J. Rumbaugh, I. Jacobson, and G. Booch. The Unified Modeling Language Reference Manual. Addison-Wesley, 1999.

3 Craig Larman, Applying UML and Patterns. Pearson Education

4 Bill Ball and Hoyt duff, Red Hat Linux and Fedora Unleashed. Pearson Education

5 Terry Quatrani, Visual Modeling with Rational Rose 2002 and UML. Pearson Education

APPENDIX

A. Workflow Engines

Name of the Workflow Engine	Available on the Web at Address
1. BEA WebLogic Integration	http://www.bea.com/
2. JBoss jBPM	http://www.jbpm.org/
3. Enhydra Shark	http://shark.objectweb.org/
4. OpenSymphony OSWorkflow	http://www.opensymphony.com/osworkflow/
5. OpenWF	http://www.openWF.com/
6.HP Changengine–Business Process Management for the Enterprise.	http://www.ice.hp.com/cyc/af/00/index.html/
7. Meteor (UGA/LSDIS)	http://lsdis.cs.uga.edu.proj/meteor/
8. OpenWFE 1.5.1	http://sourceforge.net/project/

B. Pseudo Code in C++

//******The following Pseudo code has been developed to highlight the class structure in c++ based on the Use Case diagram created in chapter 3.The details of the methods are not specified as up till now the only UML has been used and the choice of the engine for the implementation of the WorkFlow Management System has not been made and later the open WorkFlow Engine 1.5.1 was selected which is based on Java.**********//

```
//*************class administrator***************
class administrator{
//code
public:
        boolean scrutnise _thesis()
        {
        if(valid_title)
        {
                professor p;
                p.select_thesis_to_guide();
        //code
        }

        void maintain_thesis_info()
        {
        validate_user();

        //code to maintain thesis record

        }

        void maintain_course_info()
        {
        validate_user();
        //code to maintain course data
        }

        void maintain_student_info()
        {
        validate_user();
        //code to allocate thesis to student and student information
```

```
        }


        void maintain_professor_info()
        {
        validate_user();
        //code to allocate professor as guide and to maintain professor info.
        }

        void create_course_catalog()
        {
        //code to create a catalog based on the information maintained
        }


        void validate_user()
        {
        //code to validate the user personal information
        }
};


//*****************class professor**********
class professor
{
//code
public:
        void  select_thesis_to_guide()
        {
        HOD h;
        h.allocate_thesis_guide();
        //code
        }

        void select_course_to_teach()
        {
        //code to select courses to teach based on course information

        }

};



//*****************class student********
class student
```

```
{
//code
public:
        register_for_thesis()
        {
        administrator admin;
        admin.scrutnise_thesis();
        //code
        }

        resgiter_for_course()
        {
        administrator admin;
        admin.create_course();
        //code
        }
};


//*****************class HOD***********
class HOD
{
//code
public:
        allocate_thesis_guide()
        {
        administrator admin;
        admin.maintain._thesis();
        admin.maintain_student();
        admin.maintain_professor();
        //code

        }

};


//************************
void  main(void)
{
student s;
int choice
cin>>choice;
if(choice==1)
{
        s.register_for_course();
```

```
}
else
{
        s.register_for_thesis();
}
}
```