A Workflow based Academic Management System using Multi Agent Approach

SYED IMRAN JAMI and DR. ZUBAIR A. SHAIKH
Department of Computer Science,
National University of Computer & Emerging Sciences (FAST-NU),
On National Highway, Karachi
PAKISTAN
http://khi.nu.edu.pk/static/imranjami

Abstract– This paper deals with the automation of academic processes using workflow management systems. Current available workflows create lot of network traffic which leads to increase in network latency. The solution to this problem is to integrate mobile agents with workflow systems. Many researchers proposed and developed such systems but none of them consider the scenario of academic institution which has some unique features that no one showed. We in this paper propose a framework for a multi-agent based academic workflow system using Java based Aglets and XML for data representation. It uses Java Rule Engine API (JSR 094) to implement academic rules and policies of university.

Keywords: Multi agent system, academic management system, workflow model, intelligent systems,

1. Introduction

1.1. What is Work flow?

Workflow Management Coalition in its latest version defines workflow as:

”The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.”

Workflow systems deal with the automation of business process based on ruleset. A data set or work space is created, and is processed in stages at different processing points to meet business goals. All these activities are coordinated using workflow engines. Workflow engines can involve interaction either with a user or might be executed using machine resources. The automation provides huge increases in efficiency and avoid long queue delays at different processing points.

1.2. What is Work Flow Management Systems?

[1] provides a classic definition of Workflow Management Systems. It states:

“A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications”.

It also reported some advantages in implementing this system.

1. It can execute many activities in parallel which leads to reducing the time taken to handle each application and improving customer service.
2. The supervisor is only required to handle exceptions which lead to take extra burden from him because decision making is now automated.

Many workflow systems are currently available that are open source as well. These workflows lack some issues that [2] reported. Some of them are to provide systems that are scalable, have high availability, are easy to manage, and that can manage security sufficiently. In case of dynamic changes in rules and policies they require changes in whole system.

1.3. What are software agents?

Researchers [2, 3, 4] working on software agents defined it as programs that assist people and act on their behalf. This agent can be static or it can be mobile. If an agent is stationary, it executes only on the system where it begins execution. Such agents can communicate with agents on other systems using RPC or message passing system. If an agent is mobile then it can execute on any system and can transport itself from one system to another. Such agents transfer itself as objects containing code with data that are autonomous with the ability to perceive, reason and act. It has a mechanism for operating or drawing inferences from its knowledge and can interacts with other agents [4].

1.4. Agents for Workflow systems

In order to support a flow of information software agents with mobility support will be useful here, because it will provide autonomy to the activity. The mobile agents will carry the information and the task they need to perform throughout the organization without involving human users. [3] reported seven different reasons for using mobile agents. These are:

1. They reduce the network load.
2. They overcame network latency
3. They encapsulate protocols
4. They execute asynchronously and autonomously
5. They adapt dynamically
6. They are naturally heterogeneous
7. They are robust and fault tolerance.

Current workflow systems that are available commercially are running in different organizations. We, in this work considered two well known workflows namely, IBM Lotus Domino and TransparentLogics LogicBase studio. Both of these software are considered as state of art tools for managing and monitoring information workflows but both of them lacks agent based system for interacting with systems.

2. Related Work

In designing multi-agents based workflow system, people and labs [4, 6, 7, 8] worked on area that integrates workflow systems with mobile agents but these systems do not considered the unique features of academic institute's environment.

[4] presented a gluing-framework for integrating workflow processes with software agents. It proposed this framework to support cooperative software engineering processes in a better and more flexible way. The paper claimed that existing workflow systems are suited to model and support the CSE processes that are structured and repeatable, while the multi agent based systems are to model and support dynamic and cooperative processes that may involve some negotiation as well.

[6] presented an agent-based cross-enterprise Workflow Management System (WFMS) architecture which can dynamically integrate the workflows and compose a workflow execution community customized to different workflow specifications. The paper combines agents with workflows to effectively integrate cross- enterprise workflows. It claims that current workflow technologies are unable to cope with dynamic interactions and interoperability which is what an agent can provide. This work is helpful in B2B e-commerce systems.

[3] presented a workflow for office systems that independently manage workflows and personal schedules. It claims that existing workflows cannot provide this feature because of resource constraints. The “WorkWeb System” of this group is an expanded workflow system that is able to manage and control office resources. This work is done for organizations that have distributed structure.

Savarimuthu et. al. [8] presented a multi-agent based information workflow that monitors and controls the business processes. It also develops a framework for an adaptive and distributed agent based workflow system JBees, which is used to monitor and control the system based upon the data obtained through simulation. The work is proposed for businesses that have dynamically changing environment. The commercial systems for Workflow management lack controlling of the process model using agents. This makes them vulnerable in case of dynamic changes in rules and policies.

The models proposed by different researchers [4, 6, 7, 8] however integrate agents with work-flow systems but none of them considered the unique environment of academic institutions.

3. Our System

3.1. Current Scenario

The academics department of FAST-NU, is responsible for two jobs - examination and registration process. The examination process is simple and do not require automation. The registration process since last year follows some automation but it lacks the proper workflow environment. The current process works as follows:

Before the beginning of a new semester the academic department issues a list of courses offered in the semester. The offering of the courses is online process and each student needs to select the courses as per their choice. After selecting the courses they submit the request to academic department which is also an online process. Students are then required to print the request form and submit it to the respective faculty advisors. Upon receiving the forms, faculty advisor checks the courses for registration. The advisor then check the request for its success or failure based on the past performance of student by seeing its past grades, pre-requisites and GPA. This is an online process. If a student does not qualify for a course he needs to select another course(s) and starts the whole process again. If he passes, then advisor returns the form to the student with his seal and signature. Student takes this form to the accounts department for the payment of fees and other dues.

Accounts department upon receiving the fees return the form to student with its signature and seal. The student will then submit this form to academics department which after seeing approvals from concerned departments finalize the registration process using online software. After finalizing this process students are given one week time for add/drop course.

Three people/departments are involved in this exercise - Academics, accounts and advisors. Although the major process of registration is online but due to the active involvement of (human) the process took time with extra delays due to too much requests.

3.2. Our Approach

The above scenario shows that academic system differs in structure as compared to B2B or ecommerce organizations. First, it does not require to interact with outside system unlike other organizations which involves heterogeneous systems. Second, departments provide services to the students just like in public utility companies.

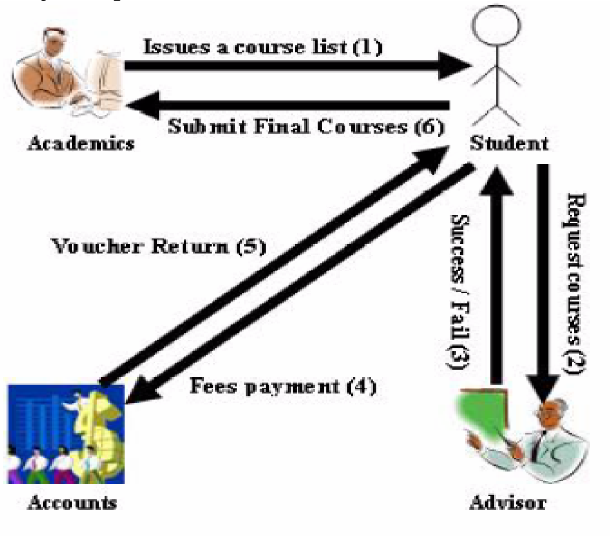

Figure 1: Current Scenario

Third, in utility companies, customers can come from any domain whereas in academics 'customers' are from closed domain - our students. Last, academic rules can be changed dynamically or it may be possible that different faculties may not want to implement same rules, thus require a rule repository with rule engine for their system.

The typical scenario shown above, require strong 'mobility' of student to disseminate information among different processing points. Moreover current system cannot process multiple requests simultaneously which leads to immense delays in registration. It also involves department supervisors to act as an interface between students and systems. This system also faces problem in implementing changes in rules and policies. With the proper workflow systems such problems can be removed by automating academic process.

The commercial systems for Workflow management lacks controlling of the process model using agents. The models proposed by different researchers integrate agents with workflow systems but none of them considered the unique environment of academic institutes.

The proposed system is working as follows:
There are four agents in this system – Academic agent to work as academic supervisor, Advisor agent to work as faculty advisor, accounts agent to work as an accounts supervisor. Student agent to work as a student, which is mobile and requires to interact with these three agents.

1. Student (human) first selects the required courses issued by the academic department and submit the request.
   (a) This submission actually invokes 'Student Agent' in the workflow system that delivers this information to the academic department.
2. At academic office, 'student agent' interacts with 'academic agent' for submitting its request.
   (a) 'Academic agent' first checks the status of the courses that include cap requirements, clashes with other courses etc.
   (b) If a student (agent) qualifies initial requirements then it replies to the student agent to go ahead to advisor.
   (c) Upon receiving the success message from academic agent, student agent then proceed to Advisor.
3. At faculty advisor's office, 'Student agent' submits its request to advisor, by interacting with 'Advisor agent'.
   (a) 'Advisor agent' checks each course request by checking the student's past performance, GPA, Grades and successful completion of pre-requisite courses.
   (b) If a student (agent) qualifies for registration, it replies to the student agent about its decision.
   (c) Upon success or failure, student agent then proceeds to accounts department.
4. At accounts department, 'student agent' submits it requests for those courses that are allowed by advisor agent.

(a) 'Accounts agent', checks the past dues if required to pay by the student (agent). It replies the student agent to pay additional dues for successful registration of courses.
(b) This agent then calculate the total fees required by the student to pay for those courses that are allowed by advisor agent.
(c) 'Accounts agent' is responsible for informing students about the amount payable by students using automated email.
(d) Upon receipt of payment by accounts department, 'accounts agent' notify pending 'student agent' to proceed back to academics department.

5. Academics department then update the states provided by the receiving student agent. The registration process gets completed here. Each agent has an access to the rule repository that contains rules and policies of university for the students.

3.3. Implementation Details

Several technologies are available to implement the proposed architecture. Two well known agent architectures are: FIPA and MASIF. FIPA do not provide mobility support in its standard whereas MASIF which is endorsed by OMG provides strong mobility support in its standard. Most of the agent systems are developed on Java environment. [2] reported this trend because of Java's support for platform independence, secure execution, dynamic class loading, object serialization and multithreaded programming.

We, in this work, use MASIF based Aglets that are Java objects move from one host to another [2]. This object can take the program with its data. Mobile agents using Aglets is useful in distributed applications where processing is migrated toward resources [2]. This type of paradigms useful for academic applications where immediate results are required on low bandwidth network. Aglet is considered as the shorthand for agent plus applet. It provides an infrastructure framework for building mobile agents based distributed applications.

Aglet is a java-based internet agent. It provides the functionality of creation, dispatch, migration and termination of agents on low bandwidth network because it uses a technique called serialization to transmit data on the heap and migrate the interpretable byte-code [2].

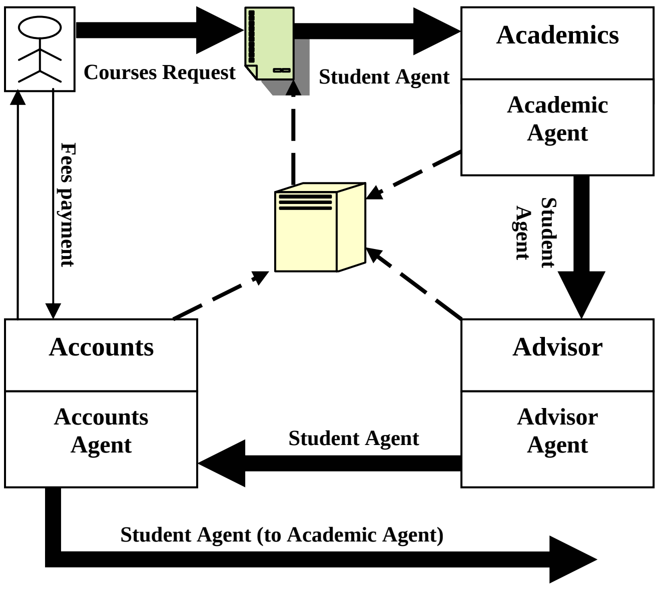

Figure 2: Our Agent based system

In our implementation framework Fig 3 the Aglet layer works in the same way as used in its architecture. It has the following components: an aglet viewer named tahiti that provide context to the agent, an aglet server and finally aglets themselves that are interacting with it. Aglet viewer is in many senses an applet viewer. It allows to create, retract, activate, deactivate and dispatch aglets. It contains a network daemon that listens for incoming aglets. It also has a security layer that protects the host from malicious agents Aglet server can host large number of aglets with large amount of data or computing resources.

On dispatching 'student' user aglet, each aglet carries XML data (state) and code for processing as an itinerary that is input by student. Student Aglets visit several hosts shown in our implementation, perform necessary communication with respective aglets at each host for computation and finally carries the result back to 'Academic' aglet for updation. Each host has its own aglet server and connected with centralized database. The student side of the application on submission of application on XML page will dispatch 'student' aglets to 'academic' aglet servers. The academic aglet server dispatches 'academic' aglet that communicate with 'student' aglet to exchange required information as shown in previous section. Getting results from academic agent, student aglet then dispatch to advisor aglet server which then dispatch 'advisor' aglet. It exchanges required information with each other and on getting results (permission etc.) from advisor agent dispatch to account aglet server. On receiving request from student aglet, account aglet server dispatches 'account' agent and they interchange information with each other. Finally student aglet dispatched to academic aglet server which stores information in student information system.

Each agent (student, advisor, academic and accounts) is represented by identifier of agent which is globally unique and immutable throughout the lifetime of the aglet. At each unit aglet is dispatched to a destination instead of retracting it from a location.

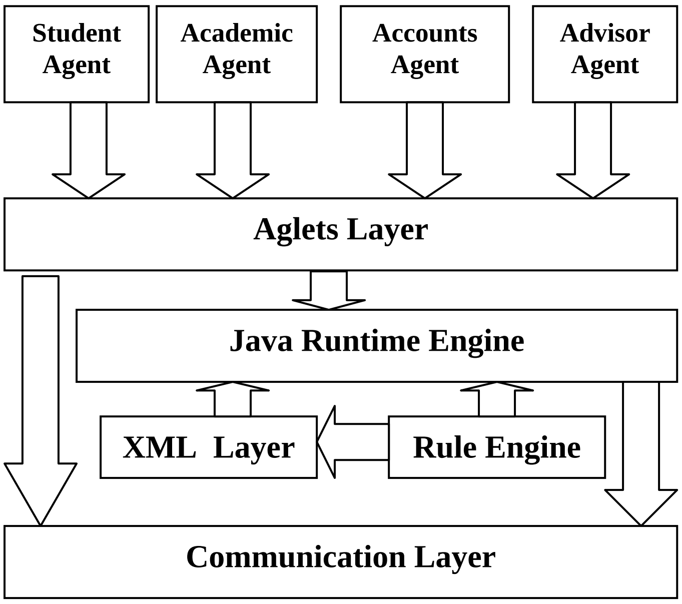

Figure 3: Implementation Framework

The communication layer of our framework deals with the interface among different aglets with each other. The aglet supports an object based messaging that is location independent, extensible and synchronous/asynchronous [2]. Aglets use message passing paradigm for communication supporting unicast, multicast and broadcast paradigms. In our framework, the communication layer provides interface among aglets using unicast message passing because message sender (e.g. student) know the identity of the receiver (e.g. advisor). This framework ensures parallel execution which is required in our system so that multiple students can submit their applications simultaneously and multiple supervisors (academic, advisor and accounts) can respond to their requests.

The aglet creates a student aglet for academic host. The student aglets are dispatched one by one to academic host. Similarly, on receiving student aglet by academic aglet server, an academic (or advisor, account) aglet is created for each of them one by one. For data representation we propose to use XML to represent information about students in the architecture. The major reason to use this for data is the availability of tools in Java to support XML. Markup languages are nowadays widely used in online data manipulation based systems. Most of the agent based systems[4, 6, 7, 9] use these languages in their architectures due to interoperability.

XML is considered as the most state of art technology in this area. [10] in his article reported several reasons to use this language for data representation.

- It is straightforward to use XML over the Internet. Users can view XML documents as quickly and easily as HTML documents.
- It supports wide variety of applications for authoring, browsing, content analysis, etc.
- It interchanges structured documents over the web based system.
- It is easy to program for processing XML documents.

It is found that most of the agent based data processing systems use XML for data manipulation tasks due to similar reasons.

In existing scenario student communicates with department's supervisors using physical forms. In our agent based system, forms are used for interaction of students for selection of courses etc. In todays interactive web based applications forms are an important part of the applications. XML in its framework provides XForms for getting users input. The XForms give the separation of the data being collected from the controls which makes it tractable by making it clear what is being submitted where. XForms have many benefits over existing HTML forms. The ease of submission process to XML helps in marshaling the submitted data to the application back-end. The received XML instance document can be processed directly by the application back-end. XForms also separates content and presentation.

Aglets is based on Java, XForms can easily be developed on Java. XForms will capture responses from student, advisor, academic and accounts and these reponses are required to transmit using Aglets.

All rulesets are specified in XML resource file. This file contains list of rules that can be accessed by aglets through Java. Issues like restricting students from registering in a course (due to low GPA or attendance etc.) are implemented by XML representation of rules. To access and extract rules, Sun Java provides Java Rule Engine API specified in JSR 94 [11, 12] for parsing rulesets that are represented using XML. In Fig 3, the layer of rule engine handles this working. It can interact with XML layer to extract the rules and with Java runtime engine for API. Aglet access the XML rule repository using Java rule engine. Java runtime layer provides this access. Rule engine evaluates and executes rules which are specified in XML as if-then statement. This layered architecure

[Fig 3] seperates implementation details of workflow with rulesets that enable our system to adapt itself with dynamic changing without any modification in source code.

JSR 94 provides Rules Administrator API [11] which is defined in the javax.rules.admin package, to load rules and associated actions as execution sets. Rule Execution set is a collection of rules. It is loaded from XML file. The API provides authorization by defining the permissions on execution sets. The 'advisor' and 'academic' agents are authorized to access this file.

4. Conclusion

The paper proposes to use Aglets platform for interaction among agents while XML for data representation. Use of Aglets is due to its support for mobility which is required in our problem. It is to be mentioned that Aglets provide weak mobility. There are some more platforms, that offer strong mobility one of which is popularly known as Grasshopper [5].

JSR 094 defines generic API support for parsing rulesets represented in XML but it does not define any standard. Research is going on to develop simple rule markup language (SRML) for standardization [12]. With our layered architecture, the system can adapt dynamic changes easily and efficiently without changing source code. Any change can be implemented in XML rule repository without disturbing Aglet and Java. This makes our system robust in a changing environment which other systems can not provide.

This work assumes that all academic process is carried out under one roof, however recent trends in educational sector suggest that many universities have campuses in different locations. Further extension of this idea can be made by making it available to all the campuses of the university. The use of agents in workflows will make WFMS scalable to many students and extensible for multi-campuses.

References:

[1] Gerhard Weiss, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, 1999

[2] Danny Lange and Mitsuru Oshima, *Programming and Deploying Java Mobile Agents with Aglets*, Addison Wesley - 1998.

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

[4] Alf Inge Wang, "Using a Mobile, Agent-based Environment to support Cooperative Software Processes", Ph.D. Thesis, Norwegian University for Science and Technology, February 5, 2001

[5] IKV. GrassHopper – The Agent Platform. web: http://www.ikv.de/products/grasshopper/

[6] Liangzhao Zeng, Anne Ngu, Boualem Benatallah Milton O'Dell, "An Agent-Based Approach for Supporting Cross-Enterprise Workflows", *Proceedings of IEEE*, 2001.

[7] Hiroyuki Tarumi, Koji Kida, Yoshihide Ishiguro, Kenji Yoshifu, Takayoshi Asakura, "WorkWeb System - Multi-Workflow Management with a Multi-Agent System", *Proceedings in ACM* 1997.

[8] Bastin Tony, Roy Savarimuthu, Maryam Purvis, Martin Fleurke, "Monitoring and controlling of a multi-agent based workflow system", *Australasian Workshop on Data Mining andWeb Intelligence (DMWI 2004)*, Dunedin. Australian Computer Society, Inc 2004

[9] Alf Inge Wang, Reidar Conradi, Chunnian Liu Y, "Integrating Workflow with Interacting Agents to support Cooperative Software Engineering", Norwegian University of Science and Technology, (NTNU), N-7491 Trondheim, Norway.

[10] Norman Walsh "A Technical Introduction to XML", http://www.xml.com

[11] Qusay H. Mahmoud, "Getting Started With the Java Rule Engine API (JSR 94): Toward Rule-Based Applications" July 26, 2005 web: http://java.sun.com

[12] "Simple Rule Markup Language (SRML)", *Technology Reports*, http://xml.coverpages.org/srml.html