

SWIFT: Scalable Workflow Management System using Mobile Agents

Amit Padalkar, Pranav Nabar, Sachin Arora, and Prasad Naik

Abstract—A *workflow* defines flow of information and control in a process. Most software solutions for workflow management systems do not model the real world scenario perfectly. Moreover, changes in the workflow result in many modifications to the application. The *Mobile Agent* paradigm not only provides a richer model for workflow systems that better resembles the real world scenario but can also be used to incorporate changes in the workflow in an elegant manner. We present a prototype of a workflow system developed using mobile agents. The workflow considered is that of a *course registration process*, used by the Indian Institute of Technology, Bombay.

Index Terms—Mobile Agents, Workflow, Workflow Systems, Course Registration System

I. INTRODUCTION

WORKFLOW defines flow of information and control in a process. A typical workflow involves multiple entities following a predefined set of rules or task specifications and co-operating towards some common goal. The specifications of a workflow system include the actions to be performed, routing information and policies that describe the organizational environment. *Workflow management* is the efficient management and execution of workflow between the involved entities for increased profitability and throughput[CGN96].

Workflow management systems automate the flow of execution in workflows. They usually decrease latency by reducing human involvement. They also allow one to dynamically define, execute, manage and modify processes. Their use facilitates isolation of the control flow of an application from domain specific logic. This separation makes it easier to modify processes without having to change the application and data structures.

An *agent* is any program that acts on behalf of a user. A *mobile agent*[GKN⁺96][HCK95] is a program, which is capable of moving across the network from node to node and perform some useful computation. The tasks that it can perform range from online shopping to real-time device control to distributed scientific computing.

Mobile agents are well suited for workflow management systems[JHS⁺99]. They follow the steps specified in a predefined process and can move to sites to gather information like a normal user. While doing this, they carry process-specific code and data thus avoiding the need to consult a central database server or the originating machine at every step. Using mobile agents workflows can be modeled so that they better resemble the real-world process. Besides

mobile agents have other advantages [FPV98][HCK95] like reduced network traffic, low network latency, disconnected operation, etc.

SWIFT is a prototype for a course registration system using mobile agents. This system models the course registration process for graduate students in Indian Institute of Technology Bombay. The system allows students to register for the courses of their choice out of the ones offered in the current semester. SWIFT runs on the *Voyager* mobile agent framework.

We start off with a detailed explanation of the course registration process followed in IIT Bombay (section II). In section III and IV, we introduce the design and implementation of our SWIFT course registration system. Section V presents a brief performance analysis of our mobile agents based workflow system and we conclude our discussion with section VI.

II. COURSE REGISTRATION PROCESS

In this section, we discuss the course registration process for graduate students at IIT Bombay. The chief entities involved in the workflow are:

- *Student*, primary user of the system
- *Academic office*, manages the course registration process
- *Faculty advisor*, acts as an advisor for the students in the department
- *Course instructors*, offer various courses to the students in the particular semester

At the beginning of every semester, each department announces a list of courses to be offered. The credits and pre-requisites for each course are also given. A student is required to fulfill a certain number of credits in every semester. Depending on these credit requirements, the student selects an appropriate number of courses. In addition to these, he/she can audit at most one course per semester. An audited course does not carry any credits.

The student collects the course registration form available at the academic office. He/She then fills the form, mentioning his/her course choices, and submits it to the concerned faculty advisor. Each department has a faculty advisor for each program. He/She scrutinizes and attests the form. Depending on the student's background and interests, he/she may suggest alternate courses also. In some cases, the students registering for a particular course might have to take the approval of the concerned instructor(s). In such cases, the course registration form has to carry the attestation of the course instructor(s) as well. The form is then submitted back to the academic office for further processing.

Amit Padalkar, Pranav Nabar, and Sachin Arora are graduate students at Kanwal Rekhi School of Information Technology, IIT Bombay. e-mail: {amitp, pranav, sachin}@it.iitb.ac.in

Prasad Naik is a graduate student at Department of Computer Science and Engg., IIT Bombay. email:prasad@cse.iitb.ac.in

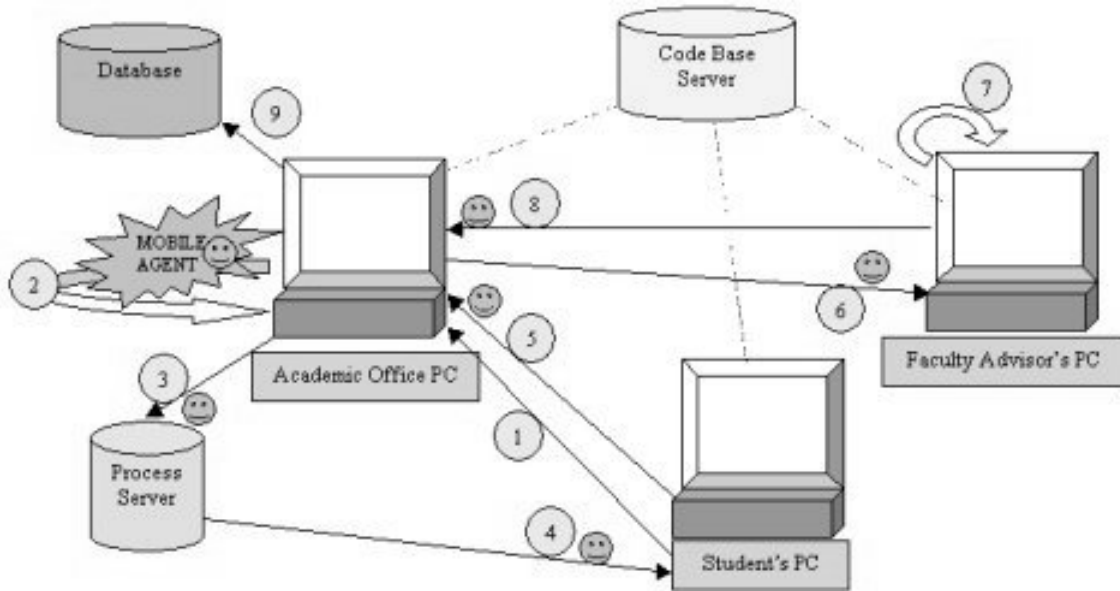


Fig. 1. Course Registration Workflow

III. SWIFT: SYSTEM DESIGN

SWIFT uses the *Dartflow model*[CGN96] for workflow management systems. Our system consists of three main components:

- *User interface*, which handles interactions with the users
- *Mobile agents*, which carry flow control information and data
- *Process server*, which stores the process logic

The user interfaces manage display of application form and verification form with relevant information on student and faculty advisor machines respectively. It also performs local validation of data wherever possible.

A single mobile agent is responsible for taking care of process instance from start to end. This agent works in conjunction with the process server that provides necessary process logic. It provides necessary support for mobility among nodes.

The process server is the most important component in SWIFT. Its job is to maintain specific process implementations and provide the agent with appropriate process implementation when requested. The process server design allows for changes in registration process to have minimal effect in the application. It also facilitates generalizing the application for different processes.

In SWIFT, a mobile agent is responsible for moving among the entities involved in the course registration process. However it does not have apriori knowledge of how to implement the process. Instead the agent requests the course registration procedure from the process server. The process server then attaches appropriate procedure to the mobile agent. This procedure knows the path to follow for carrying out course registration and the actions to take at each node. This dynamic attachment facilitates coopera-

tion between process implementation and mobile agent to execute the registration workflow.

IV. SWIFT IMPLEMENTATION AND TESTING

We implemented SWIFT using the *Voyager*[Obj] mobile agent framework and the *Java* programming language. The Voyager framework provides a facility of dynamically attaching code to an object. The attached code is called a facet. In SWIFT, a mobile agent is created by attaching a mobility facet to a normal object. This agent then attaches itself a facet implementing a course registration process. Each process facet is required to implement an interface called *IProcess*, which lists basic facilities that a process implementation must provide. A process is composed of many steps. These steps are required to implement the *IStep* interface. The facilities listed in *IProcess* interface include executing current step using *IStep*'s execute method, moving to next node, stopping the process etc. Once a registration process facet is attached to mobile agent, it travels to destinations specified in the facet and executes corresponding process step on reaching the node.

The dynamic attachment of appropriate process implementation to the mobile agent is made possible using dynamic aggregation feature of the Voyager mobile agent framework. Besides dynamic aggregation, SWIFT also utilises other features of Voyager framework like mobility, codebase server etc.

Three P-III/550 MHz systems running on Windows 2000 Professional operating system were used as test bed. These test systems using SWIFT had Voyager mobile agent execution environment running on them. These systems represented academic office PC, faculty advisor's PC and the student PC respectively. Process server and related databases were maintained on the academic office PC.

A. Course registration using SWIFT

Here we discuss the steps involved in course registration using SWIFT.

Step 1 The student requests a course registration form through a web interface.

Step 2 At the academic office, a mobile agent (MA) is spawned in response to the above request.

Step 3 The MA collects the course registration procedure from the Process Server located at the academic office.

Hereafter, the MA follows the path specified in the course registration procedure that involves the following steps:

Step 4 The MA goes to the student and displays the registration form (user interface).

Step 5 After the student has submitted the completed form, the MA captures the registration data and goes to the academic office. Here the data is validated using the student database. If data is found to be invalid, the MA returns to the student. (*Step 5a*)

Step 6 The MA then goes to the respective faculty advisor's machine and displays an Approval request form (user interface).

Step 7 The faculty advisor scrutinizes the data and digitally signs the form.

Step 8 The MA goes to the academic office where the faculty advisor's digital signature is verified. A mail is then sent to the student informing him about the status of his registration. If the faculty advisor rejects his/her course choices, the student has to personally approach him and repeat the process.

Step 9 The registration data is stored in a database to complete the process.

For simplicity, we haven't taken into consideration the cases where the student needs to take the approval of course instructor for registration. Though we can easily modify our workflow solution to include the instructor as well.

V. PERFORMANCE EVALUATION

Using the *dartflow model* for designing workflow applications based on mobile agents, we can ensure minimal changes in the system[CGN96], even for a drastic alteration in the workflow process. The use of *process server* makes the design more adaptable. For example, as a change to the original course registration workflow, we may like to introduce course instructors. All we need to do is alter the procedure in the process server to include the code for handling instructors in the mobile agent itinerary. So, the next time when the agent comes to the process server, it receives this altered procedure for the workflow. It then accordingly goes about executing the corresponding tasks without being aware of any modifications in the procedure. Thus, we obtain generality in the application.

The system uses a *Code Base Server* which has the code, that can be fetched by mobile agent on a need basis, for executing steps. This allows the agent to move to various

locations without having to carry the code for all its tasks, thus reducing the network load and utilizing bandwidth effectively. Security is an inherent problem with mobile agents. We have employed digital signatures in SWIFT for authentication. Though this method may suffice for the course registration system, it may not be adequate for critical workflow systems.

Workflow systems modeled using mobile agents closely resemble the real world workflow. This makes designing the system a simpler task and facilitates better understanding of the system. This also reduces the possibility of design errors in more complicated workflow system. The use of mobile agents in workflow systems has also been investigated by other researchers[KSD99].

VI. CONCLUSION & FUTURE WORK

We implemented SWIFT course registration system and found that the mobile agent paradigm is very useful for workflow systems. We can model real life scenarios in a better way.

For simple workflow systems, mobile agent model may not have a significant performance gain over the client-server model. But for more involved workflows, the advantages of the mobile agents in general, viz. low network load, reduced network latency, disconnected operation, dynamic adaptability, and seamless integration are prominent.

In today's world, where electronic applications such as electronic shopping and online education are ubiquitous, mobile agents can prove to be quite promising. But, security remains a major concern for critical workflow systems. More work needs to be done to make mobile agents and the host systems more secure.

ACKNOWLEDGEMENTS

We would like to thank Prof. Sridhar Iyer and Vikram Jamwal for their insightful comments and discussions on this work and paper. We would also like to thank Bhalchandra Barve for his constant support and feedback on the topic.

REFERENCES

- [CGN96] Ting Cai, Peter A. Gloor, and Saurab Nog. Dartflow: A workflow management system on the web using transportable agents. Technical Report TR96-283, Deptt. of Computer Science, Dartmouth College, 1996.
- [FPV98] Alphonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna. Understanding code mobility. In *IEEE Transactions on software engineering*, 1998.
- [GKN+96] Robert S. Gray, David Kotz, Saurab Nog, Daniela Rus, and George Cybenko. Mobile agents for mobile computing. Technical Report TR96-285, Deptt. of Computer Science, Dartmouth College, 1996.
- [HCK95] C. Harrison, D. Chess, and A. Kershbaum. Mobile agents: Are they a good idea? Available at <http://www.research.ibm.com/massdist/mobag.ps>, 1995.
- [JHS+99] Jin Jing, Karen Huff, Himanshu Sinha, Ben Hurwitz, and Bill Robinson. Workflow and application adaptations in mobile environments. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, February 1999.
- [KSD99] G. Kaiser, A. Stone, and S. Dossick. A mobile agent approach to lightweight process workflow. Technical Report CUCS-021-99, Columbia University, 1999.

[Obj] Inc. ObjectSpace. Voyager 4.0 documentation. Available at <http://support.objectspace.com/doc/index.html>.