Mobile Agents for effective structuring of large-scale distributed applications

Vikram Jamwal
KR School of IT
IIT Bombay
Mumbai, 400 076, India
vikram@it.iitb.ac.in

Sridhar Iyer
KR School of IT
IIT Bombay
Mumbai, 400 076, India
sri@it.iitb.ac.in

ABSTRACT

Large scale distributed applications typically involve a number of nodes, which may be spread over a large geographical area. The Client-Server(CS) paradigm has been found useful for designing distributed applications. However, CS approach does not scale well when the number of participating nodes increase and build complex relations with one another. Also, increase in geographical spread and unreliable network connectivity pose problems for CS implementations.

Mobile Agents (MA) is emerging as a useful paradigm for overcoming the above limitations [6]. We have designed and implemented a large distributed application, viz. distance evaluation, which uses MA as the underlying design approach. Our system provides the infrastructure for conducting computer based testing of students who may be dispersed around the globe.

In this paper we present the software engineering lessons learnt from our implementation and put forth the various issues that emerge. We show that MA paradigm can be exploited for effective structuring of large-scale distributed applications. We claim that, compared to other approaches, MA based approach gives us several advantages such as: scalability, flexible structuring, dynamic extensibility, push-pull modes of information dissemination, transparency to varying communication channels, application layer multicasting, and dynamic content delivery.

Keywords
Mobile agents, distributed application architecture

1. INTRODUCTION

Large scale distributed applications are characterized by a number of nodes, which may be distributed over a large geographical area. The various components of such an application may change their relationships with each other over a period of time; for example, a component which behaves as a client may subsequently become a server for some other components. The underlying network characteristics, such as link bandwidths and delays, may also vary over a period of time. It is difficult and cumbersome to model such applications using only traditional architectures like client-server, peer-to-peer, master-slave etc.

We argue that the MAs are particularly suitable for these large-scale applications. A MA is an autonomous software entity that can migrate between various nodes of the network and perform computations at these nodes. MA carries its state information while moving from one node to another. A MA will usually have an *itinerary*, which is a list of nodes it needs to visit, associated with it. [3] and [6] give several reasons for using mobile agents.[1] and [7] provide the motivation for using mobility to structure distributed applications.

We propose to show that software mobility in general and mobile agents in particular help in effective structuring of large-scale distributed applications. The gains are in terms of scalable and flexible architectures, and dynamically extensible applications. MAs enable both push and pull modes of information dissemination. They facilitate application level multicasting and delivery of dynamic content. Where underlying network architecture is unpredictable or frequently changing, MA based approach provides better solutions.

We have implemented Mobile Agents based system for Distance Evaluation (MADE) for computer based testing of students distributed over large areas. We have used the MA approach for designing and implementing our system. We have found that this approach yields many advantages over other traditional approaches.

In this paper we detail our experiences and the insights gained through MADE. Section 2 provides an overview of MADE. In section 3 we discuss the software engineering issues that exist in large-scale distributed applications and how they may be addressed through mobile agents. In section 4 we conclude our discussion.

2. DESIGN OF MADE

Distance evaluation of students constitutes a crucial factor for the success of distance education. Traditional Computer Based Testing (CBT) techniques, which have relied upon CS design, are now being extended to the Internet based testing [2]. These systems face drawbacks like susceptibility to network outages and network latencies. Moreover, many of these systems do not cover the full examination process (described below) and concentrate only on testing and evaluation.

We have designed and implemented a Mobile Agent based system for Distance Evaluation (MADE), which provides an integrated and comprehensive solution to evaluate students who are connected through the Internet. MADE provides support not only for testing but also for paper setting and evaluation.

In MADE, we divide the examination process into three stages: (i) examination setting, (ii) distribution and testing, and (iii) evaluation and result compilation

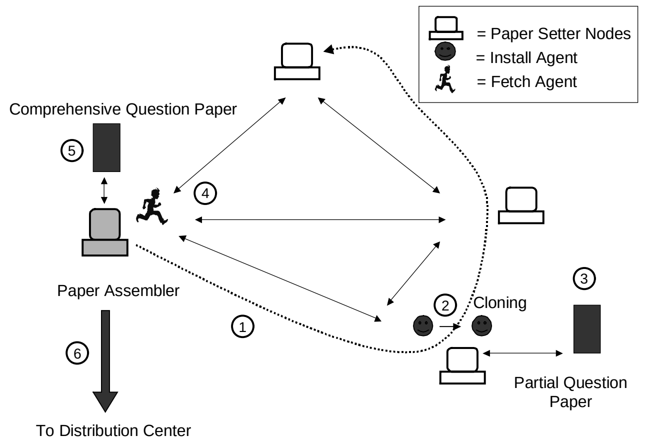


Fig 2.1 Examination Setting

2.1 Examination Setting

The examination setting process (Fig 2.1) takes place in a collaborative manner among the paper-setters who are at different remote locations. *Install Agents* are used to install the paper-setting application on their machines. An Install Agent does this by moving to a paper-setter node, cloning itself and then moving to the next node (Steps 1 and 2). Each setter prepares a partial questionnaire (Step 3). When it is time to collect these partial question papers, *Fetch Agents* are dispatched to these examiners. The itinerary for Fetch Agents is dynamically decided and they can repeatedly visit the nodes in any order (Step 4) until they have gathered all the partial question papers. The Paper Assembler node creates a final question paper based on the inputs from different examiners (Step 5). On the due date and time, this comprehensive question paper is forwarded to the distribution center.

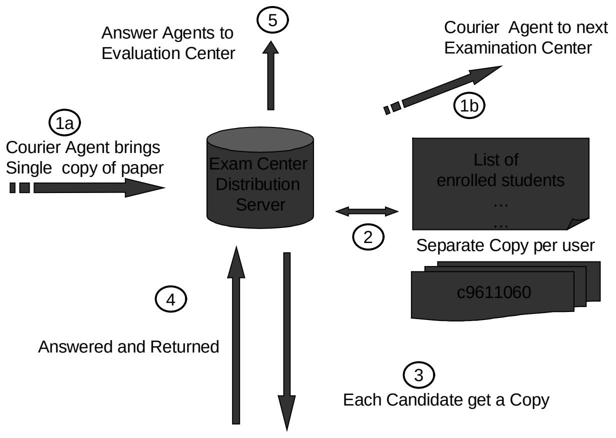

Answer Agents to Evaluation Center
5
Courier Agent to next Examination Center
1b
1a
Courier Agent brings Single copy of paper
Exam Center Distribution Server
List of enrolled students
2
Separate Copy per user
c9611060
4
Answered and Returned
3
Each Candidate get a Copy

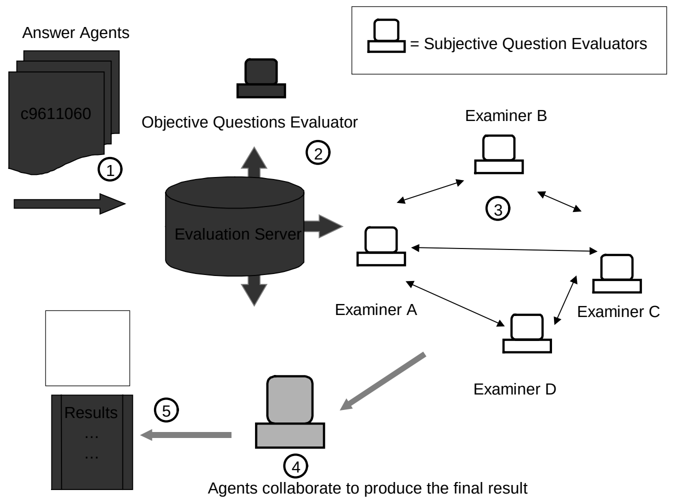
Fig 2.2 Distribution and Testing

2.2 Distribution and Testing

This stage (Fig 2.2) involves distributing the question paper to different centers, supplying the question paper to each of the enrolled students and then collecting back their answers. The question paper is dispatched to the different examination centers with the help of *Courier Agents* (Step 1a, 1b). Having finished their distribution work, the Courier Agents either get terminated or they return to their place of origin. The distribution servers at these centers have a list of candidates enrolled for that center. The distribution server creates *Question Agents,* which contain the examination paper (Step 2), and dispatches them to each student machine in the center (Step 3). The Question Agents can time themselves out after a fixed interval of time. When a student finishes answering a question paper or is timed out, the answers are given back to the distribution server of the center (Step 4). Distribution server launches an *Answer Agent* for each student answer-paper. We have segregated the Question Agents from the Answer Agents to minimize the risk of a student node tampering with the Question Agent and finding out the evaluation process details. Finally, the Answer Agents make their way to the Evaluation Center (Step 5).

2.3 Evaluation and Result Compilation

In this stage students' answers are evaluated, and the results are compiled and published (Fig 2.3). Once an Answer Agent reaches the evaluation center (Step 1), it is supplied with an itinerary of the examiners. The Answer Agent can also move to an Objective Question Evaluator (Step 2), if it possesses answers to objective questions, to get its answers evaluated. The Answer Agents move from one examiner to other, until all of the questions are evaluated (Step 3). They then move to the Publishing Center where they supply their results and where the final comprehensive results are compiled (Step 4) and published (Step 5).



= Subjective Question Evaluators
Answer Agents
c9611060
Objective Questions Evaluator
Examiner B
Evaluation Server
Examiner A
Examiner C
Examiner D
Results
…
…
Agents collaborate to produce the final result
1
2
3
4
5

Fig 2. 3 Evaluation & Result Compilation

We have implemented the above system using Voyager ORB[4] framework and have successfully tested it in an experimental setup on a campus network. In the next section we present the software engineering lessons that we learnt from our experience with MADE.

3. SOFTWARE ENGINEERING USING MOBILE AGENTS

Our experience with the MADE gave rise to the following insights on the use of MAs for effective structuring of large-scale distributed applications:

3.1 Scalable applications

MA based design enables efficient addition (or even deletion) of nodes participating in an application.

In MADE, the itinerary of various agents can be dynamically decided and new distribution center, paper-setter, examiner and student nodes can be added to the system without affecting the performance of the system. Since each of these nodes is autonomous, its addition does not unduly load any single component of the system.

3.2 Flexible structuring of applications

If the components of an application are MAs (or mobile objects), they can be placed anywhere on the network effortlessly and during run time. They are placed where they are best utilized. New components and relations can be added to or removed from the system easily.

For example, in MADE, during the paper-setting process, we use Install Agents to set up the whole collaborating infrastructure. Thus a change in system architecture will just involve supplying the new installation rules and components to the Install Agents.

Distributed applications that are based on MAs can thus be re-organized easily. This leads to improvement in crash recovery times and dynamically controlling the size and spread of application. This is in contrast to other distributed computing mechanisms, say CORBA, which are based on the traditional client-server approach, and hence possess no explicit mechanism to achieve the above.

3.3 Dynamic extensibility

MAs can be used to upgrade an application dynamically. Functionality can be thus added to or removed form the system at run-time.

In MADE, during the paper-setting stage, the Fetch Agents attach an agent object to the paper-setter's application when they move there. This attached object becomes an integral part of the paper-setter's application. We even enhance the graphical user interface of the applications using this mechanism at runtime. The paper-setter is then able to manipulate this interface object directly. After the interactions are over, the object can be detached from the application.

Dynamic extensibility, thus, can help in version control and propagation, protocol replacement, and automatic software upgrades.

3.4 Push-Pull modes

MAs use general execution environments and thus can be used to support both the push and pull modes of information dissemination.

For example, in MADE, the question papers are delivered (or pushed) to the students just-in-time and all the students taking a particular test are evaluated simultaneously. For examination setting and evaluation, a combination of both the approaches, viz. push and pull, is used.

Client-server operates principally in pull mode while MAs, depending upon the requirements, can be either summoned or dispatched by the user.

3.5 Adapting to varying communication channels

When we use MAs, the interactions become local and dependency on the network is reduced. The network does not take part in the computation process; it only helps in transfer of process and resources. Cases, where processes can work autonomously for large intervals of time, are good candidates for MA solution.

In MADE continuous connections are not required. Message exchanges are required mostly during agent transfers and rarely otherwise. In fact, student terminals can be disconnected from the main network during the period the students are being examined.

Additionally, an application that is optimized for a particular set of network characteristics will find it difficult to adjust if the network changes its configuration.

3.6 Application layer multicasting

Content carrying MAs can route themselves through the networks nodes. They cause the information to be duplicated and forwarded only at the points where it is so required.

For example, in MADE, the courier agents carry only a single copy of the question paper. The distribution servers then duplicate them depending upon the number of students enrolled.

3.7 Variety of delivered content

By using MAs to deliver the content, information content can be dynamically varied since the execution logic is coupled with the data. MADE provides for inclusion of dynamic content in question papers in the form of audio-video clips or multimedia.

We believe that the above gains in structuring applications arise mainly due to the “mobile agent based design”. It would be extremely intricate and cumbersome to provide these advantages using traditional client-server paradigms.

4. CONCLUSIONS

Most applications of mobile agents center on using MA as the representative of a user; where the MA travels around the network performing tasks on the user’s behalf. We claim that the MA paradigm is much more powerful than this and is extremely well suited for designing large-scale applications. Applications whose components have complex changing relationships and are geographically distributed would most benefit from using MA design.

We designed and implemented one such application, viz. MADE, and found that this approach gave us a clear advantage in terms of scalability, flexible structuring, dynamic extensibility, and reduced dependency on the characteristics of the underlying network. Other advantages gained were in the form of application layer multicasting, support for dynamic content, and provision for both push and pull mode of information dissemination. Additionally, we found that mobile agents readily map onto many of the real life mobile entities, resulting in more effective designs of those systems.

We are now in the process of extending MADE to support other aspects in the distance education domain such as lecture content delivery and collaborative works.

ACKNOWLEDGMENTS

Authors are thankful to Rahul Jha and Srinath Perur for their suggestions and discussions during the development of MADE.

REFERENCES

1. Alfonso Fuggetta, Gian Pietro Picco and Giovanni Vigna. "Understanding Code Mobility ", *IEEE Transactions on Software Engineering* , vol. 24(5), 1998
2. Chien Chou. Constructing a Computer-Assisted Testing and Evaluation System on the World Wide Web-The CATES Experience, in *IEEE Transactions on Education*, Vol 43, No 3, Pages 266-272, August 2000
3. Danny B. Lange. Mobile Objects and Mobile Agents: The Future of Distributed Computing, In *Proceedings of The European Conference on Object-Oriented Programming '98*, 1998
4. G. Glass, "ObjectSpace Voyager Core Package Technical Overview ", *Mobility: process, computers and agents* , Addison-Wesley, Feb. 1999
5. Jakob Hummes, Arnd Kohrs, and Bernard Merialdo. Questionnaires: a framework using mobile code for component-based tele-exams. In *Proceedings of IEEE 7th Intl. Workshops on Enabling Technologies: Infrastructure for Collaborating Enterprises (WET ICE)*, Stanford, CA, USA, June 1998
6. J. White. Mobile Agents, in *Software Agents*, J. Bradshaw (ed.), AAAI Press / The MIT Press, 1996
7. Todd Papaioannou. On Structuring of Distributed Systems, The argument for mobility, *PhD Thesis*, Loughborough University University, 2000