

The Design and Implementation of the Chronobot/Virtual Classroom (CVC) System

S. K. Chang, Xin Li, Ricardo Villamarin, Dan Lyker, Chris Bryant

Department of Computer Science

University of Pittsburgh, USA

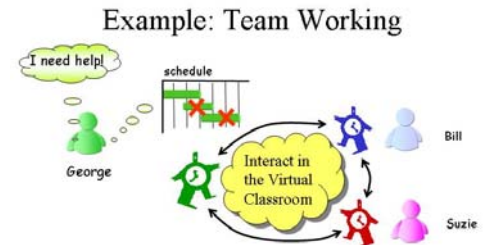
{chang, flying, rvillsal}@cs.pitt.edu, danlyker@hotmail.com, cbryant@alumni.pitt.edu

Abstract

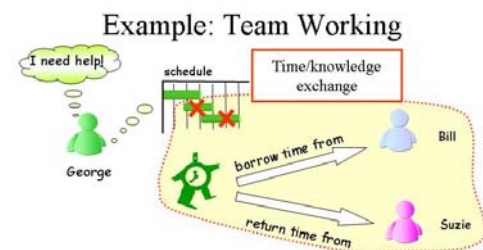
In this paper, we describe our approach to the design and implementation of the Chronobot/Virtual Classroom (CVC) system, which is a novel time/knowledge exchange platform. The system is based upon a flexible, component based architecture. The messages used by the distributed components are formulated in XML, and every component can be tested separately. The communication server is built on the basis of the Java Messaging Service (JMS) engine. Message archiving techniques are described to support unlimited message retrieving. Verified in practice, the system can provide an operational, robust platform for time/knowledge exchange in e-Learning and distance education.

1. Introduction

Chronobot/Virtual Classroom (CVC) [3] is a novel time knowledge exchange platform where any pair of users can exchange their time and knowledge. The chronobot is a time management tool for storing and borrowing time. Using Chronobot, one can borrow time from someone and return time to the same person or to someone else. The virtual classroom is a versatile communication tool that combines the functions of a web browser, chat room, white board, and multimedia display. The CVC system is an integration of the chronobot and virtual classroom that allows users to freely switch between these two applications and obtain maximum benefits from both.



(a)



(b)

Figure 1. Application of the CVC system:
(a) Communication in Virtual Classroom; (b) Time and knowledge exchange in Chronobot

For example, as illustrated in figure 1, George, Bill, and Suzie are all students who are doing a group project together in a graphics design course in which they use the CVC system to collaborate with each other. The whole project is divided into several tasks, each of which is mainly assigned to one person. George meets a problem in his task and cannot solve it by himself. So he interacts with Bill and Suzie in the virtual classroom, and eventually they help him out. However, in order to keep workload even among teammates, George has to put in efforts either in the past or in the future to help Bill and Suzie. The chronobot serves as a platform for them to do such time and knowledge exchanges which could have significant value for many applications. Many more

interesting scenarios can be found in [3].

In this paper, we describe our approach to the design and implementation of the CVC system. Aiming at a large-size, reliable, and scalable communication system, we have designed a flexible, client/server based system architecture. The functionality of servers and clients are broken down and encapsulated into components. The messages exchanged among components are formulated in a platform-independent format (XML), and every component can be separately tested using a customized testing tool, (the remote control). The most important server – the communication server – is designed and implemented based on an open source Java Messaging Service (JMS) engine, ActiveMQ [1]. Additionally, a message archiving mechanism allows users to retrieve messages within any time range.

The rest of this paper is organized as follows: the overall architecture of the CVC system is described in Section 2. In Section 3, we discuss message formulation and testing of the distributed components in the system. The communication servers and client tools are described in detail in Section 4 and 5, respectively. Message archiving is discussed in Section 6, followed by a brief conclusion in Section 7.

2. System Architecture

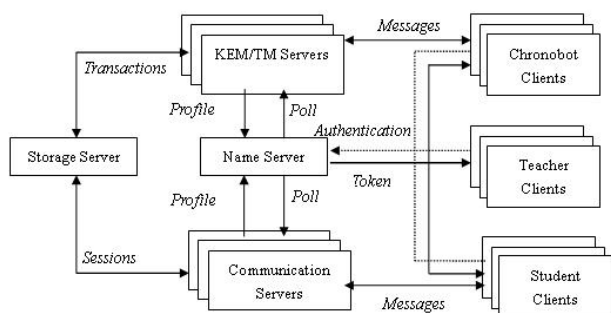


Figure 2 System Diagram

The CVC system is illustrated in Figure 2. Basically, the system follows the client/server architecture. The clients (e.g., chronobot, teacher, and student clients) provide user interfaces interacting with different kinds of users. The time/knowledge exchange and related

functions are implemented by services from the different servers (e.g., name server, KEM/TM (Knowledge Exchange Management/ Time Management) server, communication server, and storage server). Following is a brief description of these servers and clients:

- Chronobot Client: a tool for chronobot users to exchange time and knowledge.
- Student Client: a tool for students to attend the classes in the virtual classroom.
- Teacher Client: a tool for teachers to attend classes in the virtual classroom. Besides the basic communication functions found in the student client, it is equipped with some administrative functions (e.g., permission manipulation and student grades management).
- Name Server: a server providing naming services for all clients and dynamically managing the working load of the communication servers and KEM/TM servers.
- Communication Server: a server providing communication services for virtual classroom clients (i.e., teacher and student clients). The messages for learning activities in the virtual classroom are all relayed by the communication server and are recorded as *learning sessions*, or simply *sessions*.
- KEM/TM Server: a server managing time and knowledge exchange for chronobot clients. The transactions on time/knowledge are all performed in the KEM/TM server, which are recorded as *time/knowledge exchange transactions*, or simply *transactions*. For more detail about the KEM/TM server, please refer to [2, 6].
- Storage Server: a server storing the *time/knowledge exchange transactions* from KEM/TM servers, and the *learning sessions* from communication servers.

In the entire system, there is only a single instance of the name server, which is required to be known by all clients and other servers. Multiple instances of the communication and KEM/TM servers can work at the same time. When these servers start up, they will register their profiles (address, capabilities, current work load, etc.) with the name server. The name server

will dynamically assign tokens to the teacher, student and chronobot clients after the proper authentication. In order to balance the workloads among the servers, periodically the name server polls messages to the communication and KEM/TM servers to fetch their status.

3. Message Formulation and Component Testing

```

(a) <?xml version="1.0" standalone="yes" ?>
<!-- Generated by Virtual Remote 1.0 -->
<Msg>
  <Head>
    <MsgID>0</MsgID>
    <Description>Login Information</Description>
  </Head>
  <Body>
    <Item>
      <Key>FirstName</Key>
      <Value>xin</Value>
    </Item>
    <Item>
      <Key>LastName</Key>
      <Value>li</Value>
    </Item>
    <Item>
      <Key>IsTeacher</Key>
      <Value>True</Value>
    </Item>
    <Item>
      <Key>passwd</Key>
      <Value>xini12</Value>
    </Item>
  </Body>
</Msg>

(b) <?xml version="1.0" standalone="yes" ?>
<!-- Generated by Virtual Remote 1.0 -->
<Msg>
  <Head>
    <MsgID>1</MsgID>
    <Description>Result of Authentication</Description>
  </Head>
  <Body>
    <Item>
      <Key>Success</Key>
      <Value>True</Value>
    </Item>
    <Item>
      <Key>ServerIP</Key>
      <Value>127.0.0.1</Value>
    </Item>
  </Body>
</Msg>

```

Figure 3 Messages for Authentication Component in Name Server (a) Incoming message; (b) Outgoing message

The servers and clients in the CVC system are designed based on components, which are encapsulated software elements with specific functionalities [4]. For example, the name server is designed and implemented by three components, i.e., an authentication component, a profile retrieving component, and a workload balancing component. Every component has an interface designed to achieve maximum flexibility and reliability and may be configured by incoming and outgoing messages in XML. These XML messages are partially described in [5]. For example, the authentication component in the name server may be configured with the incoming and outgoing messages shown in Figure 3.

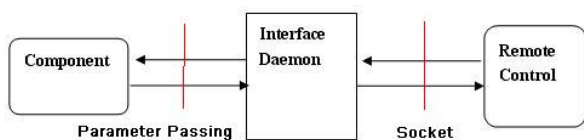


Figure 4 Testing a Component Using Remote Control

One of the most important advantages of component-based design is that the components can be

tested separately without implementation of the whole system. For this purpose, a customized testing tool named *remote control* was designed and implemented. Shown in Figure 4, the remote control can simulate any incoming messages via a programming language-independent interface (i.e., Socket). At the same time, it can receive any outgoing message transmitted by the component.

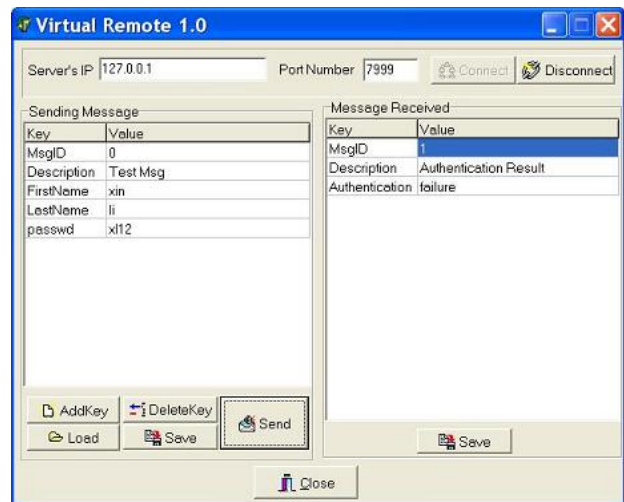


Figure 5 Remote Control (with Testing Results on Authentication Component)

Using the remote control, the functions of components can be easily verified before they are integrated into the real system. For example, Figure 5 shows the remote control with the testing results on the authentication component of the name server. A tester can send the authentication component messages containing different pairs of user names and passwords. From the server responses received by the remote control, it is very easy to check whether the component works as expected.

4. Communication Server

The communication server is one of the most important servers in the CVC system. It supports the message transmission (either broadcast or peer-to-peer) for the whole system. The communication medium used is HTTP based using an open source Java Messaging Service (JMS) engine called ActiveMQ [1]. Using HTTP protocol makes the communication stateless, and the ActiveMQ messaging allows for a persistent messaging protocol.

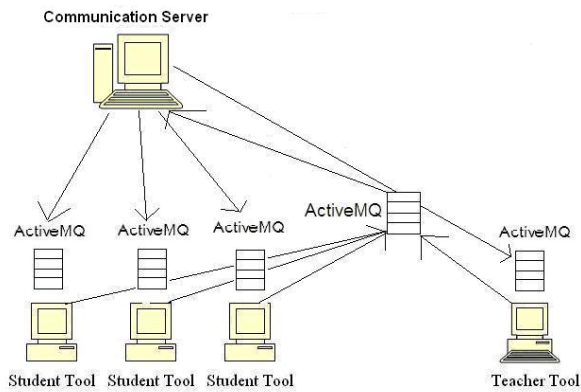


Figure 6 System Diagram of Communication Server

Figure 6 shows the system diagram of the communication server. Each student/teacher tool (i.e., client) writes to the same upload queue. The communication server reads all messages from this queue and reacts accordingly, writing all messages to the student/teacher tools to individual queues. A Java servlet is used to create an HTTP interface to a queue. This needs to be in place to take HTTP posts and HTTP gets from all ActiveMQ servers and clients. For more information about ActiveMQ refer to [1].

5. Client Tools

There are three client tools used in the CVC system: 1) the student client, 2) the teacher client, and 3) the chronobot client. Each is described in detail in the following sections:

5.1 Student Client

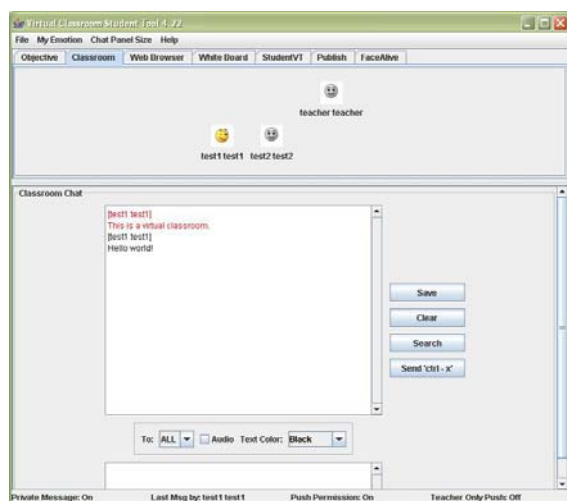


Figure 7 Student Client

The student client is designed for students in the

virtual classroom and is illustrated in Figure 7. The students (represented by emotive icons) can join a virtual classroom and exchange information in the form of text messages, web pages, sketches, and audio/video clips. The emotive icons express the feelings of a user during a learning session.

5.2 Teacher Client

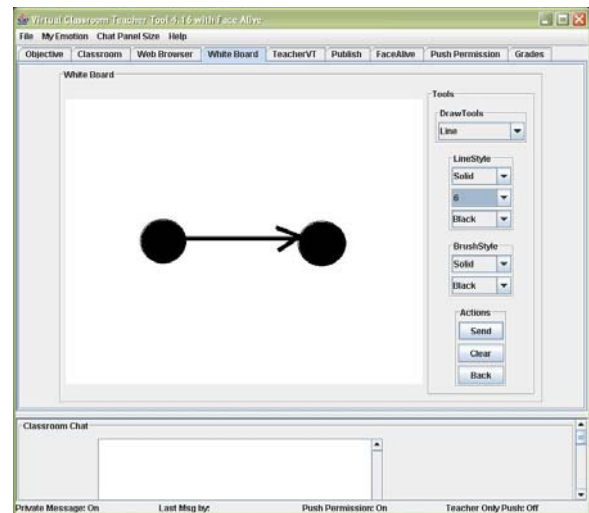
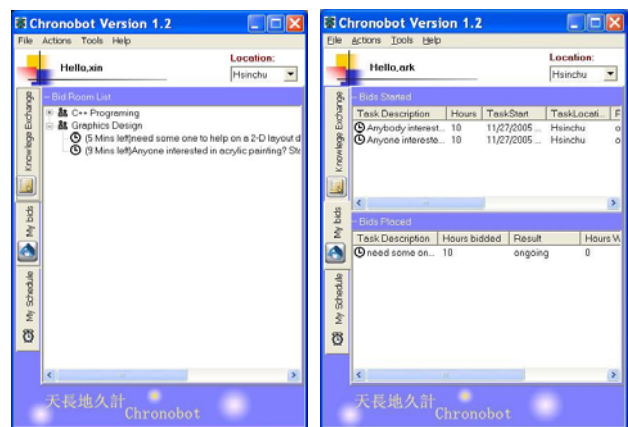


Figure 8 Teacher Client

The teacher client is designed for teachers in the virtual classroom and is illustrated in Figure 8. In addition to all the functions as available in the student client, the teacher client also includes certain administrative functions, including permission manipulation and student grade management.

5.3 Chronobot Client



(a)

(b)

Figure 9. An example of chronobot, (a): bidding room for time knowledge exchange; (b): transaction records.

The chronobot client (illustrated in Figure 9) is designed for users in the chronobot system. Basically, the chronobot client is a tool to exchange time and knowledge through bidding processes. Users can start new bids in different bidding rooms (figure 9(a)). A simple example of such bidding is “I need someone to help me on a 2-D layout design for 5 hours”. Users can respond to these requests by placing their own bids. For example, a user can respond with “I can contribute 2 hours on it”. The details about these transactions are saved into each user’s record (figure 7(b)).

6. Message Archiving

The time/knowledge exchange transactions and learning sessions accumulated in the CVC system are the most valuable assets for further knowledge acquisition (e.g., user profiling and ontology construction). Following is a brief description of the techniques used to store and retrieve transactions and sessions.

6.1 Message Storage

This is conceptually a very simple operation:

1. The server pulls messages from the main queue regularly;
2. The server stores the messages in a special table with the following basic (high-level) structure (Figure 10).

Course	From	To	Content	Time Stamp
		{ALL, <specific user ID>}	{set of fields}	

Figure 10 Message Structure

The timestamp field stores the date and time where the message was *received* by the server, not the client’s clock’s time since it can happen that the clocks are not synchronized. This serves to maintain consistency.

6.2 Message Retrieval

The sequence of operations to retrieve messages from the storage is described below:

1. The user authenticates as usual, and after that, s/he selects the course s/he wants to retrieve messages from;
2. The user asks the system to show the retrieve message history dialog (Note that at this time, this option is only available after the user has enrolled a course);

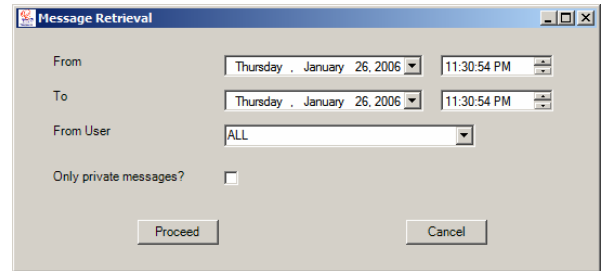


Figure 11 Message Retrieving Dialog

3. The user selects a range of dates, user IDs (of the course participants), and whether s/he wants to retrieve private messages only;
4. The client pushes a special message in the main queue that indicates that the server must perform a message retrieval;
5. Once the server receives the retrieval message, it queries the database using the criteria specified by the user;
6. All the messages found (if any) are posted in the specific queue of the user who requested the history;
7. If no message was found, a special “not found” message is posted in the client’s queue;
8. Once the client retrieves the search result from the user’s queue, it either shows the messages in the message panel, or, in the case that no message was found, the user is notified accordingly;

7. Conclusions

In this paper, we described our design and implementation of the CVC system, which is a novel time/knowledge exchange platform. A flexible, client/server based system architecture was described in which the clients and servers are all designed based on components that can be tested separately using our customized testing tool (remote control). The communication server is designed and implemented

using ActiveMQ. The client tools for the users of the virtual classroom and the chronobot were described as well, as were the techniques to store and retrieve time/knowledge exchange transactions and learning sessions. Verified in practice, the system can provide an operational, robust platform for time/knowledge exchange in e-Learning and distance education.

8. Acknowledgements

This research is supported in part by the Industry Technology Research Institute (ITRI) and the Institute for Information Industry (III) of Taiwan. We would like to thank Christopher Santamaria for the design and implementation of virtual classroom Java clients, and too many others to name here for testing our CVC system.

References:

- [1] Active MQ, <http://www.activemq.org/>, accessed on March 4, 2006.
- [2] S. K. Chang, Anupama Kapoor, Ganesh Santhanakrishnan and Chirag Vaidya. The Design and prototyping of the chronobot system for time and knowledge exchange. *The International Journal of Distance Education Technologies (JDET)*, 3(3), April, 2005.
- [3] S. K. Chang. A Chronobot for Time and Knowledge Exchange. In *Proceeding of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 3–10, Taipei, Taiwan, Jul. 2005
- [4] G. T. Heineman and W. T. Councill. *Component Based Software Engineering: Putting the Pieces Together*. Addison Wesley, 2001.
- [5] Minxin Shen. The Service Interaction Protocol for the Chronobot/Virtual Classroom (CVC) System. In *Proceeding of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 16–18, Taipei, Taiwan, Jul. 2005
- [6] E. Y. Shih and W. H. Yeh. The Implementation of Chronobot Engine. In *Proceeding of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 12–15, Taipei, Taiwan, Jul. 2005