Business Negotiation Model through Computer Mediated Negotiation

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

Computer Science

under the Supervision of

DR. NITIN CHANDERWAL

By

AAYUSH MITTAL(111301)

to

Jaypee University of Information and Technology
Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled "**Business Negotiation through Computer Mediated Communication**", submitted by AAYUSH MITTAL in partial fulfilment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Dr. Nitin Chanderwal

Associate Professor

Signature

Date: 27/05/2015

Acknowledgement

On the completion of my work scheduled for this phase of the project titled **"Business Negotiation through Computer Mediated Communication"** I would like to thank my supervisor **Dr. Nitin Chanderwal** for his able guidance, valuable suggestions and constant encouragement.

With her help I was able to complete the research work required for the completion of this phase of the project.

27/05/2015 Aayush Mittal

TABLE OF CONTENTS

LIST OF FIGURES

Abstract

With the globalization of the world economy, it is imperative that managers, both present and future, be sensitive to differences in intercultural business communication. In particular, the context of electronic commerce leads to an increasing use of email in negotiating deals, which has been so far almost exclusively the domain of face to face (FTF) or at best telephone, but not of computer-mediated communication (CMC), such as email. How to prepare for it by training, such as in the educational setting of Schools of Management and Industrial Engineering? How to analyze special effects of email on major negotiation strategies, such as "*win-win*" and "*put yourselves in the shoes of the other party*"? Psycholinguistic act analysis has proven to be a reliable and valid way to test hypotheses on intercultural communication effects. A safe first step to explore its applicability to email is to analyze it in notice writing with the only difference of using a pen instead of a keyboard as a kind of simulated email which is non FTF: people do not see each other. Such simulated email seems to require its toll by asking more indirect language. Is this always efficient? Sellers use significantly more *"you"* than buyers, which is a good sales technique. In sum, this simulated email seems to make it difficult to get involved in the other party's needs by using cooperative speech acts and second personal pronoun use. Special training, such as the one of this study could handle this handicap. Finally the results of this study lead to suggestions for future work, such as controlled studies with adequate sampling to compare CMC and FTF in different order of succession and simulation and real life with negotiation experience as a source of variation. In this study, linguistic analysis appeared to be a very useful tool to describe and verify negotiation strategies in a simulated email setting.

Index Terms – *Computer mediated vs. face to face communication, negotiation strategy, electronic discourse, speech act analysis, use of personal pronouns, mono- vs. intercultural effects.*

Chapter-1

1.1 Introduction to CMC in Business Negotiation

Computer-mediated communication (**CMC**) is defined as any human communication that occurs through the use of two or more electronic devices. While the term has traditionally referred to those communications that occur via computer-mediated formats (e.g., instant messaging, email, chat rooms), it has also been applied to other forms of text-based interaction such as text messaging. Research on CMC focuses largely on the social effects of different computer-supported communication technologies. Many recent studies involve Internet-based social networking supported by social software.

1.2 Scope of the field

Scholars from a variety of fields study phenomena that can be described under the umbrella term of CMC. For example, many take a sociopsychological approach to CMC by examining how humans use "computers" (or digital media) to manage interpersonal interaction, form impressions and form and maintain relationships. These studies have often focused on the differences between online and offline interactions, though contemporary research is moving towards the view that CMC should be studied as embedded in everyday life. Another branch of CMC research examines the use of paralinguistic features such as emoticons, pragmatic rules such as turn-taking and the sequential analysis and organization of talk, and the various sociolects, styles, registers or sets of terminology specific to these environments. The study of language in these contexts is typically based on text-based forms of CMC, and is sometimes referred to as "computer-mediated discourse analysis".

The way humans communicate in professional, social, and educational settings varies widely, depending upon not only the environment but also the method of communication in which the communication occurs, which in this case is through computers or other information and communication technologies (ICTs). The study of communication to achieve collaboration—common work products—is

termed computer-supported collaboration and includes only some of the concerns of other forms of CMC research.

Popular forms of CMC include e-mail, video, audio or text chat (text conferencing including "instant messaging"), bulletin boards, list-servs and MMOs. These settings are changing rapidly with the development of new technologies. Weblogs (blogs) have also become popular, and the exchange of RSS data has better enabled users to each "become their own publisher".

1.3 Characteristics

Communication occurring within a computer-mediated format has an effect on many different aspects of an interaction. Some of these that have received attention in the scholarly literature include impression formation, deception, group dynamics, disclosure reciprocity, disinhibition and especially relationship formation.

CMC is examined and compared to other communication media through a number of aspects thought to be universal to all forms of communication, including (but not limited to) synchronicity, persistence or "recordability", and anonymity. The association of these aspects with different forms of communication varies widely. For example, instant messaging is intrinsically synchronous but not persistent, since one loses all the content when one closes the dialog box unless one has a message log set up or has manually copy-pasted the conversation. E-mail and message boards, on the other hand, are low in synchronicity since response time varies, but high in persistence since messages sent and received are saved. Properties that separate CMC from other media also include transience, its multimodal nature, and its relative lack of governing codes of conduct. CMC is able to overcome physical and social limitations of other forms of communication and therefore allow the interaction of people who are not physically sharing the same space.

The medium in which people choose to communicate influences the extent to which people disclose personal information. CMC is marked with higher levels of self-disclosure in conversation as opposed to face-to-face interactions. Self disclosure is any verbal communication of personally relevant information, thought, and feeling which establishes and maintains interpersonal relationships. This is due in part to visual anonymity and the absence of nonverbal cues which reduce concern for losing

positive Face. According to Walther's (1996) Hyperpersonal communication Model, computer-mediated communication is valuable on providing a better communication and better first impressions . Moreover, Ramirez and Zhang (2007) indicate that computer-mediated communication allows more closeness and attraction between two individuals than a face-to-face communication.

Anonymity and in part privacy and security depends more on the context and particular program being used or web page being visited. However, most researchers in the field acknowledge the importance of considering the psychological and social implications of these factors alongside the technical "limitations".

1.4 Benefits

The nature of CMC means that it is easy for individuals to engage in communication with others regardless of time or location. CMC allows for individuals to collaborate on projects that would otherwise be impossible due to such factors as geography. In addition, CMC can also be useful for allowing individuals who might be intimidated due to factors like character or disabilities to participate in communication. By allowing an individual to communicate in a location of their choosing, CMC call allow a person to engage in communication with minimal stress. Making an individual comfortable through CMC also plays a role in self-disclosure, which allows a communicative partner to open up more easily and be more expressive. When communicating through an electronic medium, individuals are less likely to engage in stereotyping and are less self-conscious about physical characteristics. The role that anonymity plays in online communication can also encourage some users to be less defensive and form relationships with others more rapidly.

Chapter-2

2.1 Inception of the Idea

According to statistics, freelancing websites prove to be a robust and affective CMC model for Business negotiations. Thus, a software is to be constructed based on the features embedded in them.

2.2 Freelancing Marketplace

Freelance marketplaces (crowdsourcing or outsourcing marketplaces) are websites that match buyers and sellers of services provided via the internet. Service providers, or sellers, create a profile where they include a description of the services which they offer, examples of their work and in some cases information about their rates. Buyers register and complete a basic profile, they then post projects outlining their requirements. Buyers will then bid for these projects on a fixed price or hourly basis.

2.3 Services

Services offered by online marketplaces may include:

- Web Design / Internet marketing
- Graphic Design / Presentations / Multimedia
- Illustration / Cartooning / Painting / Sculpting
- Marketing / Advertising / Sales / PR
- Engineering / CAD / Architecture
- Networking / Hardware
- Legal Services
- Fashion / Interior design / Landscape
- Enterprise resource planning (ERP)/ Customer relationship management (CRM) Implementation
- Programming / Software / Database Development
- Writing / Editing / Translation
- Sales / Telemarketing
- Strategy consulting
- Management consulting
- Photography / Videography
- Finance and Accounting
- Broadcasting

\

2.4 E-lancing

E-lancing, sometimes referred to as **e-labour**, refers to the recent trend of commending and taking freelancing work through so-called e-lancing websites. E-lancing websites are hubs where employers place tasks, which freelancers from around the world bid for.

2.5 Benefits

Advantages

- Flexible working hours
- You can work anywhere
- You can choose the job you like most

Disadvantages

- Risk of non-payment
- You can get easily distracted
- Quality of work might be worse
- No stable income

Chapter-3

3.1 Java Servlet

A **Java servlet** is a Java programming language program that extends the capabilities of a server. Although servlets can respond to any types of requests, they most commonly implement applications hosted on Web servers. Such Web servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET.

3.2 Introduction

Servlets are most often used to:

- Process or store data that was submitted from an HTML form.
- Provide dynamic content such as the results of a database query
- Manage state information that does not exist in the stateless HTTP protocol, such as filling the articles into the shopping cart of the appropriate customer

Technically speaking, a "servlet" is a Java class in Java EE that conforms to the Java Servlet API, a standard for implementing Java classes which respond to requests. Servlets could in principle communicate over any client–server protocol, but they are most often used with the HTTP protocol. Thus "servlet" is often used as shorthand for "HTTP servlet". Thus, a software developer may use a servlet to add dynamic content to a web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets can maintain state in session variables across many server transactions by using HTTP cookies, or URL rewriting.

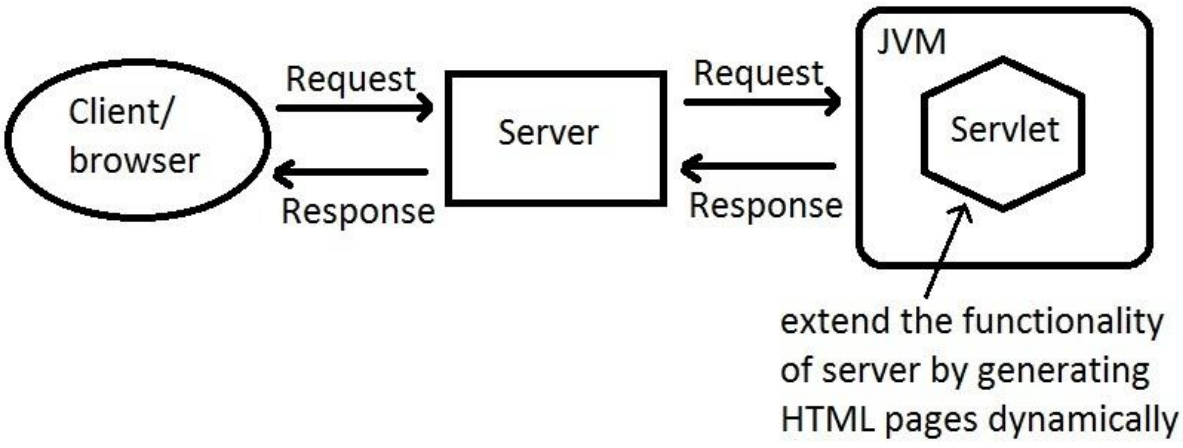
To deploy and run a servlet, a web container must be used. A web container (also known as a servlet container) is essentially the component of a web server that

interacts with the servlets. The web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

The Servlet API, contained in the Java package hierarchy javax.servlet, defines the expected interactions of the web container and a servlet.

A Servlet is an object that receives a request and generates a response based on that request. The basic Servlet package defines Java objects to represent servlet requests and responses, as well as objects to reflect the servlet's configuration parameters and execution environment. The package javax.servlet.http defines HTTP-specific subclasses of the generic servlet elements, including session management objects that track multiple requests and responses between the web server and a client. Servlets may be packaged in a WAR file as a web application.

Servlets can be generated automatically from Java Server Pages (JSP) by the JavaServer Pages compiler. The difference between servlets and JSP is that servlets typically embed HTML inside Java code, while JSPs embed Java code in HTML. While the direct usage of servlets to generate HTML (as shown in the example below) has become rare, the higher level MVC web framework in Java EE (JSF) still explicitly uses the servlet technology for the low level request/response handling via the FacesServlet.



3.3 Compared with other web application models

The advantages of using servlets are their fast performance and ease of use combined with more power over traditional CGI (Common Gateway Interface). Traditional CGI scripts written in Java have a number of performance disadvantages:

- When an HTTP request is made, a new process is created each time the CGI script is called. The overhead associated with process creation can dominate the workload especially when the script does relatively fast operations. Thus, process creation will take more time for CGI script execution. In contrast, for servlets, each request is handled by a separate Java thread *within* the web server process, thereby avoiding the overhead associated with forking processes within the HTTP daemon.
- Simultaneous CGI request will load the CGI script to be copied into memory once per request. With servlets, there is only one copy that persists across requests and is shared between threads.
- Only a single instance answers all requests concurrently. This reduces memory usage and eases the management of persistent data.
- A servlet can be run by a servlet container in a restrictive environment, called a sandbox. This is similar to an applet that runs in the sandbox of the web browser. This enables restricted use of potentially harmful servlets.[3] CGI programs can of course also sandbox themselves, since they are simply OS processes.

Technologies like FastCGI and its derivatives (including SCGI, WSGI) do not exhibit the performance disadvantages of CGI incurred by the constant process spawning. They are, however, roughly as simple as CGI. They are therefore also in contrast with servlets which are substantially more complex.

3.4 Life cycle of a servlet

Three methods are central to the life cycle of a servlet. These are init(), service(), and destroy(). They are implemented by every servlet and are invoked at specific times by the server.

- During initialization stage of the servlet life cycle, the web container initializes the servlet instance by calling the init() method, passing an object implementing the javax.servlet.ServletConfig interface. This configuration object allows the servlet to access name-value initialization parameters from the web application.
- After initialization, the servlet instance can service client requests. Each request is serviced in its own separate thread. The web container calls theservice() method of the servlet for every request. The service() method determines the kind of request being made and dispatches it to an appropriate method to handle the request. The developer of the servlet must provide an implementation for these methods. If a request is made for a method that is not implemented by the servlet, the method of the parent class is called, typically resulting in an error being returned to the requester.
- Finally, the web container calls the destroy() method that takes the servlet out of service. The destroy() method, like init(), is called only once in the lifecycle of a servlet.

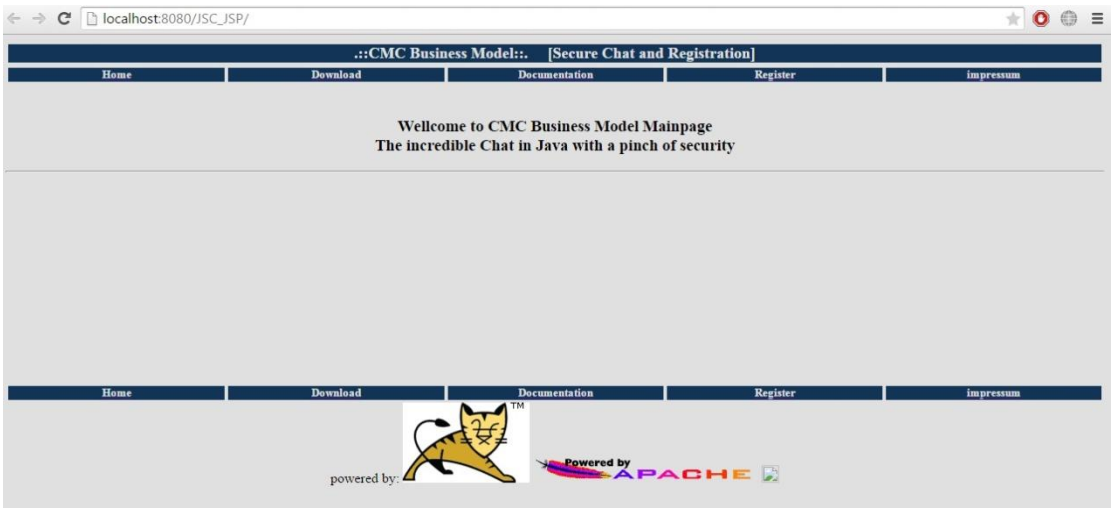
The following is a typical user scenario of these methods.

1. Assume that a user requests to visit a URL.
    - The browser then generates an HTTP request for this URL.
    - This request is then sent to the appropriate server.
2. The HTTP request is received by the web server and forwarded to the servlet container.
    - The container maps this request to a particular servlet.
    - The servlet is dynamically retrieved and loaded into the address space of the container.
3. The container invokes the init() method of the servlet.
    - This method is invoked only when the servlet is first loaded into memory.

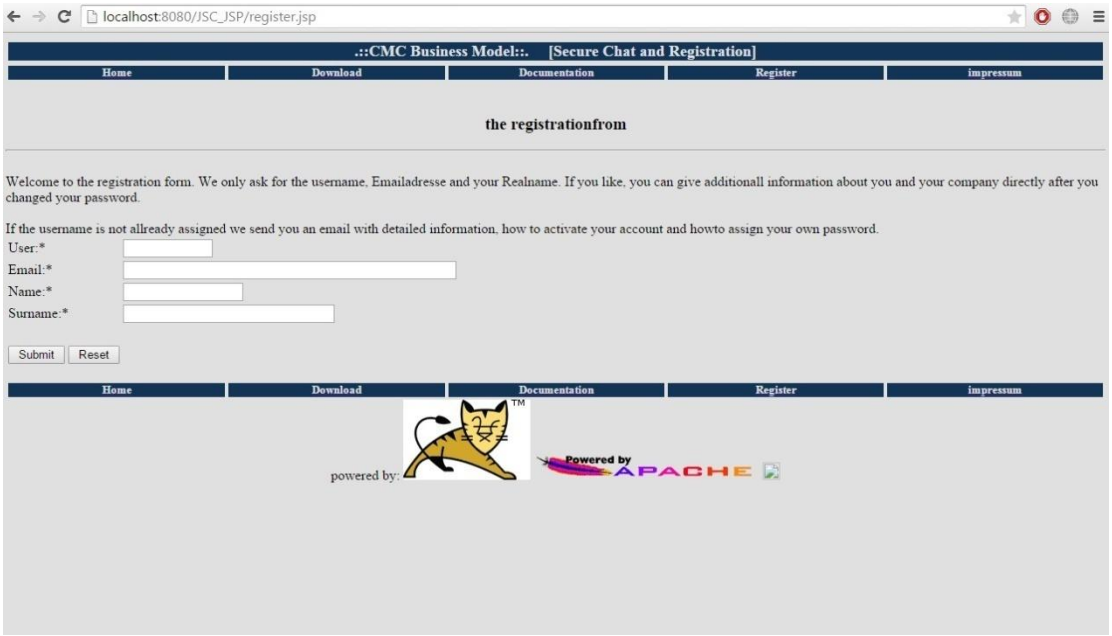
- It is possible to pass initialization parameters to the servlet so that it may configure itself.

4. The container invokes the service() method of the servlet.
   - This method is called to process the HTTP request.
   - The servlet may read data that has been provided in the HTTP request.
   - The servlet may also formulate an HTTP response for the client.
5. The servlet remains in the container's address space and is available to process any other HTTP requests received from clients.
   - The service() method is called for each HTTP request.
6. The container may, at some point, decide to unload the servlet from its memory.
   - The algorithms by which this decision is made are specific to each container.
7. The container calls the servlet's destroy() method to relinquish any resources such as file handles that are allocated for the servlet; important data may be saved to a persistent store.
8. The memory allocated for the servlet and its objects can then be garbage collected.

3.5 Implementation



Sample page for Java chat generated by a Java Servlet



Page for User Registration for the Secure Java chat Application

Chapter-4

4.1 Java Chat

Generally speaking, the job of any server is to provide a centralized service. However, there are many different ways of providing services, and many different ways to structure the communications.
Chat is roughly described as a connection-oriented service, because a user establishes a connection and maintains that connection, sending and receiving text for the duration of the session.
This is in contrast to the Web, where the protocol is (at least in theory) transactional – the browser asks for a page, and the server sends it; the connection is then closed. (In practice, the connection is kept open and reused, but this is more a speed-optimization than a structuring metaphor.)

4.2 Java/Swing programming

Swing is a rapid GUI development tool that is part of the standard Java development kit. It was primarily developed due to the shortcomings of the Abstract Windows Toolkit (AWT). For example, Swing'sJButton class enhances the AWT Button class to allow not only text, but images on the button. In addition, all Swing components support assistive technologies.

4.3 Swing Components

I am only going to go over a few basic Swing components:

- JFrame
- JPanel
- JButton

If you know how to use these basic components, using the others is simple. Usually, while making a GUI-based application, you instantiate a JFrame and choose its layout. Then you put one or more JPanels in the JFrame if you want to. JPanels also have different layout options like JFrames do. After that, you add other components.

4.4 Thread Issues

Going from AWT to Swing results in a little issue with threads. All AWT classes are thread safe; Swing's classes are not. A little knowledge about threads is necessary to completely understand the reason for the complexity of using threads. Sun's Java Tutorial covers this topic very well, so you might want to take a peek there. Consider an object that has been instantiated in one thread. Now this object needs to be modified by two or more threads. Data can easily be corrupted if two threads try to modify the same object at the same time. Of course, you know there is no such thing as "the same time" when it comes to threads since the operating system switches between the various threads to make them seem like they are running concurrently. However, if one thread is interrupted while it is modifying an object and a second thread is initiated so that it begins to modify that object, data can get corrupted.

There are a couple of ways to get around this problem in Java. Java allows a way to enforce that no thread modifies the same object that your thread is modifying through the use of the synchronize keyword. You can read up about it in Sun's Java tutorial since it is out of the scope of this tutorial. Synchronizing an object before modifying it blocks any other threads attempting to access the synchronized object until your synchronize block is done. AWT's methods synchronize on their corresponding object instantiation, which prevents the objects (eg. buttons, panels, etc.) on the screen from being corrupted while they are being drawn. However, this can block your thread if it is trying to access the AWT object. This just slows things down.

4.5 Swings and Threads

Swing, as I mentioned above, is not thread-safe. This means that most of its methods do not synchronize with the internal state of the Swing object. Only a few functions that are marked as thread-safe in the Java documentation are safe to execute from anywhere. The programmer is now given the responsibility of making sure that there are no thread conflicts. This can be done by making sure that all the modifications to the Swing objects happen in the same thread. All the events - such as button clicks and others - are handled on the thread known as the **event-handling thread**. This thread is also where all the component painting onto the screen is done. Hence, any modifications to the visible Swing components can be made in any event handling routine.

4.6 Socket Programming

The endpoint in an interprocess communication is called a socket, or a network socket for disambiguation. Since most communication between computers is based on the Internet Protocol, an almost equivalent term is *Internet socket*. The data transmission between two sockets is organised by communications protocols, usually implemented in the operating system of the participating computers. Application programs write to and read from these sockets. Therefore, network programming is essentially socket programming.

4.7 UDP Connections

With UDP, computer applications can send messages, in this case referred to as *datagrams*, to other hosts on an Internet Protocol (IP) network without prior communications to set up special transmission channels or data paths. UDP is suitable for purposes where error checking and correction is either not necessary or is performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.[1] If error correction facilities are needed at the network interface level, an application may use the Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP) which are designed for this purpose.

4.8 TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication language or protocol of the Internet. It can also be used as a communications protocol in a private network (either an intranet or an extranet). When you are set up with direct access to the Internet, your computer is provided with a copy of the TCP/IP program just as every other computer that you may send messages to or get information from also has a copy of TCP/IP.

TCP/IP is a two-layer program. The higher layer, Transmission Control Protocol, manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. The lower layer, Internet Protocol, handles the address part of each packet so that it gets to the right destination. Each gateway computer on the network checks this address to see where to forward the message. Even though some packets from the same message are routed differently than others, they'll be reassembled at the destination.

TCP/IP uses the client/server model of communication in which a computer user (a client) requests and is provided a service (such as sending a Web page) by another

computer (a server) in the network. TCP/IP communication is primarily point-to-point, meaning each communication is from one point (or host computer) in the network to another point or host computer. TCP/IP and the higher-level applications that use it are collectively said to be "stateless" because each client request is considered a new request unrelated to any previous one (unlike ordinary phone conversations that require a dedicated connection for the call duration). Being stateless frees network paths so that everyone can use them continuously. (Note that the TCP layer itself is not stateless as far as any one message is concerned. Its connection remains in place until all packets in a message have been received.)

Many Internet users are familiar with the even higher layer application protocols that use TCP/IP to get to the Internet. These include the World Wide Web's Hypertext Transfer Protocol (HTTP), the File Transfer Protocol (FTP), Telnet (Telnet) which lets you logon to remote computers, and the Simple Mail Transfer Protocol (SMTP). These and other protocols are often packaged together with TCP/IP as a "suite."

Personal computer users with an analog phone modem connection to the Internet usually get to the Internet through the Serial Line Internet Protocol (SLIP) or the Point-to-Point Protocol (PPP). These protocols encapsulate the IP packets so that they can be sent over the dial-up phone connection to an access provider's modem.

Protocols related to TCP/IP include the User Datagram Protocol (UDP), which is used instead of TCP for special purposes. Other protocols are used by network host computers for exchanging router information. These include the Internet Control Message Protocol (ICMP), the Interior Gateway Protocol (IGP), the Exterior Gateway Protocol (EGP), and the Border Gateway Protocol (BGP).

4.9 Implementation

4.9.1 System Requirements

Server System:

Each computer on which Java programs in sufficient running speed. Somehow everything from a PII and more than 64MB RAM. A network card should also be present as well as a functioning TCP / IP protocol.

Software:

Java 1.3.1 or higher

JavaMail API 1.2 or later

MySQL 3.23 and higher (3:23 Recommended)

MySQL Connector / J 2 or greater (2 recommended)

Jakarta-tomcat-4.0.5 (all versions from 3.2.1 should go)

A functioning mail server or MTA (postfix, sendmail, etc)

Client System:

Any computer which is capable of a graphical user interface for to make available. Again, should a PII with more than 64MB RAM is sufficient and of course, a network card with a functioning.

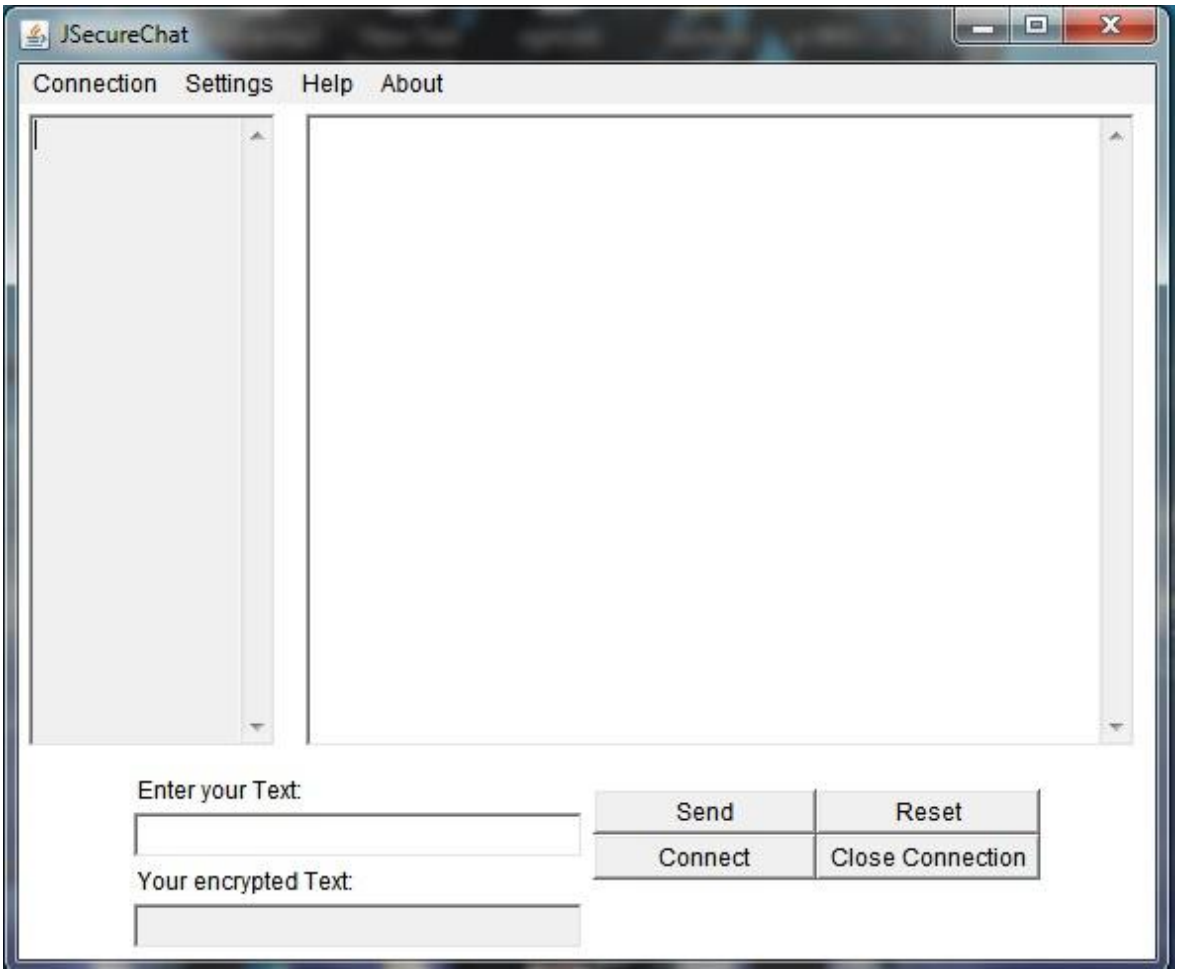
TCP / IP protocol

Software:

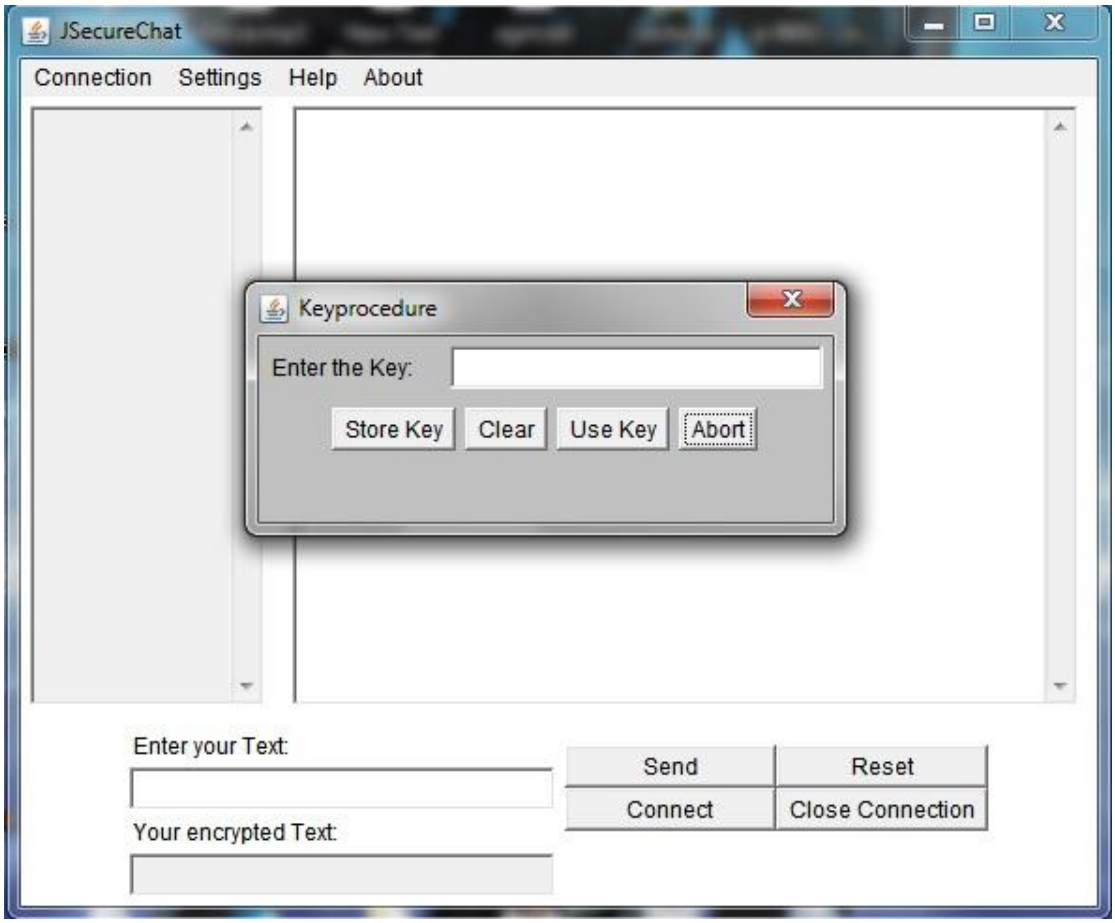
Java 1.3.1 or higher

OS with GUI

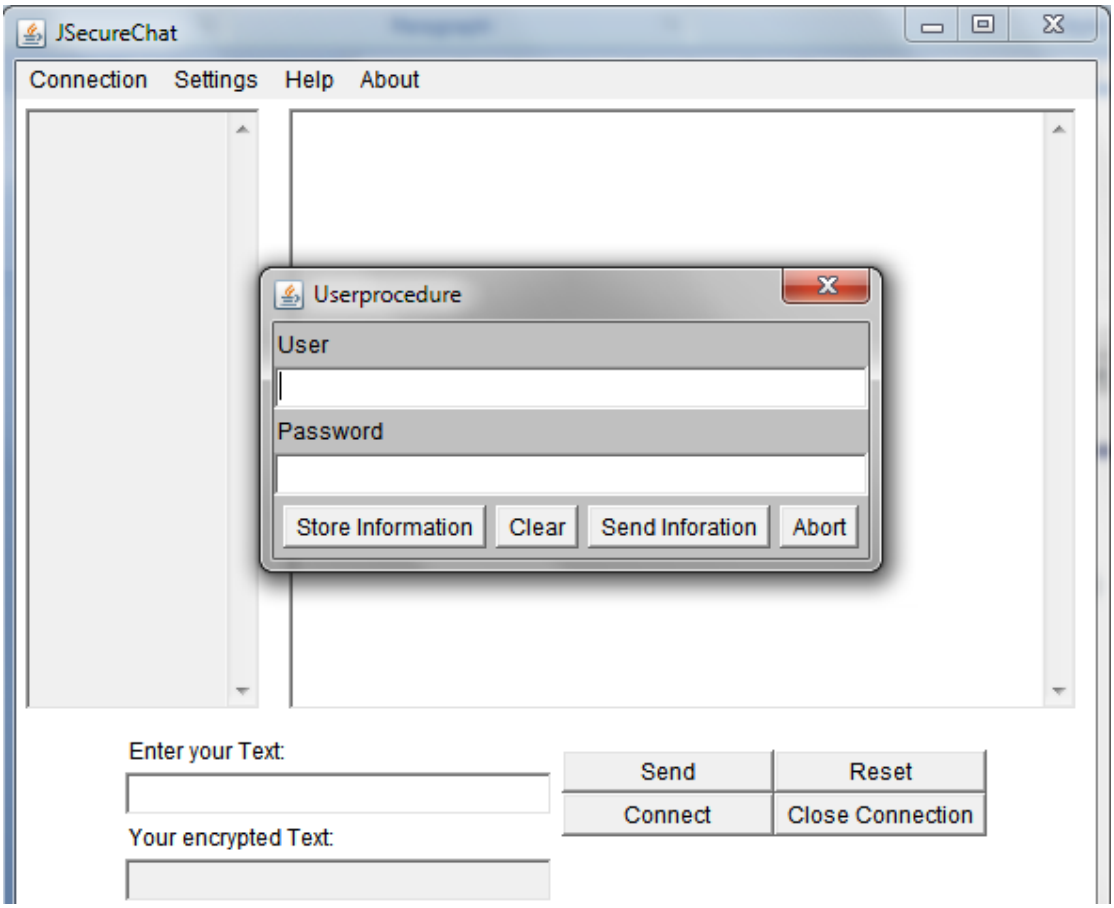
4.9.2 Running Software Screenshots



Working Secure Java Chat Application



Option to Add the encryption key (XOR Encrypted)



User Information used during online registration to be used in the chat application

Client Side:

```
public jsecurechat()
{
   /** Right, we need quite a few Buttons **/
   connect = new Button("Connect");
   /** And these Buttons need to do something, so they get ActionListeners **/
      connect.addActionListener(this);
   send = new Button ("Send");
     send.addActionListener(this);
   reset = new Button ("Reset");
     reset.addActionListener(this);
   closeconnect = new Button("Close Connection");
     closeconnect.addActionListener(this);

   /** Some TextFields for the User Information **/
   tf1 = new TextField("",29);
      tf1.setEditable(true);

   tf = new TextField("", 29);
      tf.setEditable(false);
      /** Because we want to use the Enter-Key to send data.
      We need to add an Listener and tell him what to do **/
      tf1.addKeyListener(new KeyAdapter()
```

```
    {
      public void keyPressed(KeyEvent e)
      {
       if (e.getKeyCode()==KeyEvent.VK_ENTER)
       {
         try
         {  /** This is defined further down **/
            sendMessage();
         }
         catch (Exception err) {chat.append("\ndoesnotwork\n" + err + "\n");}


         tf1.setText("");
         tf1.requestFocus();
       }
       }
       }
      );



encr = new Label("Your encrypted Text: ");
text = new Label("Enter your Text: ");


/** Now there are Millions of Panels, to get the graphics nice.
```

```
There might be a more elegant way to handle this, but it works **/
textlines = new Panel();
textlines.setLayout(new GridLayout(2,1));
   textlines.add(encr);
   textlines.add(tf);



anotherpanel = new Panel();
anotherpanel.setLayout(new GridLayout(2,1));
   anotherpanel.add(text);
   anotherpanel.add(tf1);



anotherpanel2 = new Panel();
anotherpanel2.setLayout(new GridLayout(2,1));
   anotherpanel2.add(anotherpanel);
   anotherpanel2.add(textlines);

butt = new Panel();
butt.setLayout(new GridLayout(3,2));
   butt.add(send);
   butt.add(reset);
   butt.add(connect);
```

```
        butt.add(closeconnect);



    paneltext = new Panel();
    setLayout(new BorderLayout());


    paneltext.add("Center", anotherpanel2);
    paneltext.add("East", butt);


    panelchat = new Panel();
    chat      = new TextArea(20, 57);
    panelchat.add(chat);


    upanel = new Panel();
    userarea = new TextArea(20, 15);
    userarea.setEditable(false);
    upanel.add(userarea);



/** The Menu Connect that handles all subjects concerning the
connection to the Server **/


     menue = new MenuBar();
```

```
Connection    =    new Menu("Connection");


/** It contains the following Items: **/
/** Key: here you can change the key that is used by the encryption**/
   key    =    new MenuItem("Key");


/** User: Here you enter your Username and Password **/
   user    =    new MenuItem("User");


/** Connect: You can, and have to specify the IP-No for the Server **/
   connectwith    =    new MenuItem("Connect with");


/** Port: You can, and have to specify the Port-No to contact the Server **/
   portno    =    new MenuItem("Port Number");



/** The Menu Settings. Here you can change the graphics...if this
function will ever be implemented. Feel welcome to do this **/
Settings    =    new Menu("Settings");
    graphics    =    new MenuItem("Change the graphics");


/** The Menu Help. You won't need it! **/
```

```
Help    =    new Menu("Help");
   helpdocu    =    new MenuItem("Help");

/** The Menu About **/
About    =    new Menu("About");
   about    =    new MenuItem("About:");

/** Put it all together... **/
   Connection.add(key);
   Connection.add(user);
   Connection.add(connectwith);
   Connection.add(portno);
   Settings.add(graphics);
   Help.add(helpdocu);
   About.add(about);

/** ...and make it visible **/
   menue.add(Connection);
   menue.add(Settings);
   menue.add(Help);
   menue.add(About);
```

```
    /** Assigns the WindowListener to jsecurechat **/
    this.addWindowListener(this);

    /** Assigns the ActionListeners to the Menu **/
    key.addActionListener(this);
    user.addActionListener(this);
    connectwith.addActionListener(this);
    portno.addActionListener(this);
    graphics.addActionListener(this);
    helpdocu.addActionListener(this);
    about.addActionListener(this);

    /** Assigns the Layoutmanager **/
    setLayout(new BorderLayout());

    /** Assigns the Menu **/
    setMenuBar(menue);

    /** Tells the Layout Manager where to put the staff **/
    add("West", upanel);
    add("South", paneltext);
    add("Center", panelchat);
}
```

```
/** This is the part that handles the behaviour of the Frame
(close window, end program) It is used by the WindowListener**/
  public void windowClosing(WindowEvent event)
   {
      if (event.getID() == Event.WINDOW_DESTROY)
      System.exit(0);
    }

   public void windowOpening       (WindowEvent event){};
   public void windowClosed        (WindowEvent event){};
   public void windowActivated     (WindowEvent event){};
   public void windowDeactivated  (WindowEvent event){};
   public void windowDeiconified   (WindowEvent event){};
   public void windowIconified      (WindowEvent event){};
   public void windowOpened        (WindowEvent event){};


/** This part uses the ActionListener and defines the actions performed by
the differend signals **/
public void actionPerformed(ActionEvent event)
{
```

```
/** If you hit the menue key, another window will pop-up **/
if(event.getSource()== key)
{   if ( keyframe == null){keyframe = new KeyactionDialog(this, "Keyprocedure");}
    keyframe.show();
}

if(event.getSource()== user)
{   if (udig == null){udig = new UseractionDialog(this, "Userprocedure");}
    udig.show();
}

if(event.getSource()== connectwith)
{   if (cond == null){cond = new ConnectionDialog(this, "Store Server IP
procedure");}
    cond.show();
}

if(event.getSource()== portno)
{   if (pn == null){pn = new PortnumberDialog(this, "Store Port Number procedure");}
    pn.show();
}

if(event.getSource()== helpdocu)
```

```
{   if (hp == null){hp = new HelpDialog(this, "You need Help?");}
    hp.show();
}

if (event.getSource() == send)
{
  sendMessage();
}

if (event.getSource() == reset)
{
   tf1.setText("");
}

if (event.getSource() == connect)
{
 startClient();
}

if (event.getSource() == closeconnect)
{
   try {
         socket.close();
```

```
        } catch (Exception err) {System.err.println(err.toString());}
    }
}

/** you need to put this in a thread, otherwise it
will block the whole system **/
private void startClient()
{
 Thread t = new Thread(this);
 t.start();
}

      public void run()
      {
       System.out.println("run");
       /** It insists on a try-catch thing, because theres quite a bit
       that can (and will) go wrong **/
       try
       {
        String server = new String();
       /** Create a Reader to access all the User-Data **/
        is=new ObjectInputStream(new FileInputStream("ChatData.dat"));
        data =(PrivateData)is.readObject();
```

```
   /** Read all the Data from the File ChatData.dat and put
   them in Strings **/
    server = data.adress;
    checkuser = data.user;
    checkpassword = data.password;
    checkkey = Integer.toString(data.key);
    port = data.po;

   /** Close the Reader **/
    is.close();

   /** Open the Socket with the User-Information
   Server = Server-IP or adress and Port = Port-Number **/
    socket = new Socket(server, port);

   /** Open the Streams for the Input and the Putput **/
    in = new DataInputStream(socket.getInputStream());
    out = new DataOutputStream(socket.getOutputStream());

 try
 {
/** First of all, we need to tranfer the username and the password
```

```
authentification at the server for the **/
 out.write((checkuser + ":" + checkpassword +"\r\n").getBytes());
 }
  catch (IOException es) {System.err.println(es.toString());}


    while(true)
    {
      System.out.println("blah");
      int c;
       /** As long as there is data coming at the Input Stream...**/
         while ((c = in.read()) != -1)
         {
        /**...put it in the String line **/
          line = line + ((char)c);


          System.out.print((char)c);


            if(c == 10 )
            {
              /** The Filter for the participating Usernames **/
               if(line.startsWith("///"))
               {
                userarea.replaceRange("",0,300);
```

```
//blah  ::
//blah.textarea.flush();
while(line.indexOf("::") != -1)
{
 System.out.println("geht");

 // slashe weg
 line = line.substring(3, line.length());

 // substrings, getrennt durch ::
 try
 {
  String user = line.substring(0,line.indexOf("::"));
  line = line.substring(line.indexOf("::") -1, line.length());
  userarea.append(user+ "\r\n");
  System.out.println(user);
 }
 catch(Exception e){}
 //upanel.userarea.add(user);

 }
}
else
```

```
{ /** Some more filters to get rid of the funny signs that
   are included in the data and that we don't want to show up **/
if(!line.startsWith("null"))
{
 if(!line.startsWith("Anmeldung"))
 {
  String line1 = line.substring(0, line.indexOf(":"));
  String line2 = line.substring(line.indexOf(":") +2, line.length()-2);
  int key = Integer.parseInt(checkkey);
  System.out.println("key " + checkkey);
  String inmsg=new String("");

    for (int i = 0; i < line2.length(); ++i)
    {
     inmsg=inmsg +((char)(line2.charAt(i) ^ key));
     System.out.println("en:" +((line2.charAt(i)) ^ key));
    }
      jsecurechat.chat.append(line1 +": " + inmsg);
   }
      else
      {
       jsecurechat.chat.append(line);
      }
```

```
                }
                }
               line = "";
             }  }
          }}
        catch (ClassNotFoundException e){System.err.println(e.toString());}
        catch (IOException es) {System.err.println(es.toString());}
      }

private void sendMessage()
{
 try
 {
  /** Create message **/
   String message =  (tf1.getText() + "\r\n");
  /** and grab the key **/
   int key = Integer.parseInt(checkkey);
   String outmsg = new String("");

   for (int i = 0; i < message.length(); ++i)
   {
      /** encrypt the message and use the key for it **/
      outmsg = outmsg +((char)(message.charAt(i) ^ key));
```

```
            System.out.println("en:" +(byte)((message.charAt(i)) ^ key));
        }


        out.write((outmsg + "\r\n").getBytes());
        tf.setText(outmsg);


      } catch (IOException err) {chat.append("\ndoesnotwork\n" + err + "\n");}
        tf1.setText("");
        tf1.requestFocus();
    }
}
```

Conclusion

I have used the best possible working model on Business negotiation through CMC and tried to implement it into a software. The different parts of my software would be integrated soon as I am already having troubles trying to run single modules (Secure Java chat). I assure you this project would be one of a kind and would kinder new inceptions in terms of CMC in Business negotiations.

Future Works

Until now, I have tried to generate web pages through JavaServlets and have almost successfully completed a Secure Java Chat application. My future works shall include integrating a Video Chat functionality and a way to integrate secure payment transactions for both the employee and the employer.

REFERENCES

1. McQuail, Denis. (2005). *Mcquail's Mass Communication Theory*. 5th ed. London: SAGE Publications.
2. Thurlow, C., Lengel, L. & Tomic, A. (2004). Computer mediated communication: Social interaction and the internet. London: Sage.
3. Walther, J. B. (1996). Computer-mediated communication: Impersonal, interpersonal, and hyperpersonal interaction. Communication Research, 23, 3-43.
4. Walther, J. B., & Burgoon, J. K. (1992). Relational communication in computer-mediated interaction. Human Communication Research, 19, 50-88.
5. Haythornthwaite, C. and Wellman, B. (2002). The Internet in everyday life: An introduction. In B. Wellman and C. Haythornthwaite (Eds.), The Internet in Everyday Life (pp. 3-41). Oxford: Blackwell.
6. Skovholt, K., Grønning, A. and Kankaanranta, A. (2014), The Communicative Functions of Emoticons in Workplace E-Mails: :-). Journal of Computer-Mediated Communication, 19: 780–797. doi: 10.1111/jcc4.12063
7. http://www.cise.ufl.edu/~amyles/tutorials/tcpchat/
8. ibm.com/developerWorks