DESIGN AND IMPLEMENTATION OF COLLECTIVE BARGAINING SUPPORT SYSTEM (CBSS) - A WEB-BASED NEGOTIATION SUPPORT SYSTEM

By
SUARGA, BSc, M.Sc, MMath

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
Doctor of Philosophy

McMaster University

A WEB-BASED NEGOTIATION SUPPORT SYSTEM

DOCTOR OF PHILOSOPHY (1997)
(Management Science/Information System)

McMaster University
Hamilton, Ontario

TITLE : Design and Implementation of Collective Bargaining Support System (CBSS) - a Web-based Negotiation Support System.

AUTHOR: Suarga, BSc. (Gadjah Mada), M.Sc. (Asian Institute of Technology), M.Math (Waterloo)

SUPERVISOR : Professor Yufei Yuan

NUMBER OF PAGES: viii, 189

ABSTRACT

Negotiation is an important part of the life in many organizations. Many works have been undertaken to build tools for conducting negotiations, such as theoretical models and computerized support systems. Some of the existing computerized support systems provide suggestions and solution alternatives to the users. Other systems support the negotiation process, including: support for communications, structuring the process, and managing the documentation.

The advent of the World Wide Web as widely available networking platform, the Client/Server as a computing architecture, and Java language as a programming language for the Web, creates new and promising ways to conduct negotiations as an alternative if face-to-face negotiations are not possible. This article describes the design, implementation, and validation of a computerized process support for negotiation, a Web-based Collective Bargaining Support System (CBSS). This system is implemented on Microsoft's Windows 95 environment, written in Java (Sun's JAVA SDK version 1.0), and accessed through the Web.

Simulated union-management contract negotiations were conducted in an experiment to test the validity of CBSS as a negotiation tool, and to investigate the effectiveness and the efficiency of this negotiation support system. The data analysis (of questionnaires distributed during the experiment) shows that CBSS is a valid alternative, both when face-to-face negotiation is not possible and when it is combined with face-to-face negotiations. It is also concluded from the experiment that, although CBSS is perceived to be slower than face-to-face negotiation, it is an effective tool for negotiators, and it does not negatively affect bargaining outcomes.

Keywords: Negotiation Support System, Group Decision Support System, Java Applications.

ACKNOWLEDGEMENTS

I would like to thank the members of the Supervisory Committee: Dr. Y. Yuan, Dr. N. P. Archer, and Dr. J. Rose, for their assistance, guidance, and supervision in the preparation of this dissertation.

I am most grateful to my Supervisor, Dr. Y. Yuan, for encouraging me to undertake my research, and for his help and guidance throughout the years of my study at McMaster University. My thanks to him here can never fully express my appreciation for his empathy and time.

I also gratefully acknowledge the financial assistance received from Canadian International Development Agency (CIDA) throughout the years of my study in Canada.

I would also like to thank my colleagues, Melina S. Head, Christine Manuel, and Parbudyal Singh, for their help during the software testing and bargaining simulations. For all students who participate in the experiments, thank you.

Finally, to my wife Murni, my sons Reza and Nizar, thank you very much for your patience, support, and encouragement.

Table of Contents:

List Of Figures:

List of Tables:

Chapter 1. Introduction.

The potential of conflict is particularly visible in formal settings such as group decision making and negotiation situations, although it may also happen in informal settings such as family gatherings. Conflicts arise in the normal course of events between individuals and groups. Unresolved conflicts can cause serious damage to all parties involved. For instance, unresolved conflicts between management and unions may trigger and cause losses in production, markets and profit to the organization, and unemployment and wage losses to union members.

In a group decision making environment, conflict is an important aspect of group dynamics, because ."... lack of conflict can lead to groupthink, predisposing catastrophic decision..." [Miranda and Bostrom, 1994]. Conflict in an organization need not be eliminated but it must be managed properly, because "...improperly managed conflict can lead to hostility between groups, poor motivation and morale, diminished performance and poor decisions" [Rahim ,1985].

Group interaction research identifies two types of group tasks which involve conflicts among the members of groups [McGarth, 1984]. The first type is a decision making group with members who share common interests and goals. The members of this group may find

themselves in conflicts because they have different sets of criteria which they used in making a decision. The second type involves two parties which have different interest and goals, such as a buyer and a seller. These parties are in conflict, because the interests and goals may contradict, but they still need to cooperate to reach an agreement.

Conflict resolution is a procedure used to search for a solution in order to bring a final consensus which is accepted by the parties in conflict. Many methods have been developed for resolving conflicts. These methods can be viewed either as an art or as a science. They may also be based on different theories such as psychology and human behaviour theories, optimization, and game theory.

As an art, conflict resolution is based on "interpersonal skills, the ability to convince and be convinced, the ability to employ a basketful of bargaining ploys and the wisdom to know when and how to use them" [Raiffa 1982]. As a science, "...conflict resolution can be viewed as a systematic analysis for solving problems that cause the conflict" [Raiffa 1982].

Negotiation is one way to solve conflicts where participants attempt to reach mutually acceptable agreements. In labour-management relations, a conflict between management and employee union is resolved through negotiation between their representatives in a process known as **Collective Bargaining**. Collective bargaining is an interactive decision-making process between unions and employers. The collective bargaining process is not easy to accomplish successfully. In many cases it takes a long time, is costly, and possibly creates strikes or work stoppages. Many theories and models have been established in order to

explain and to assist this bargaining process, and recently computer and information technology have also been applied to assist the process.

Interest in providing computer based system support for negotiation has been growing since the 1960s [Nyhart and Goeltner, 1987]. More recently, some specific Group Decision Support Systems (GDSS) have been developed to support group decision making, such as Electronic Meeting Systems (EMS) and Negotiation Support System (NSS). An EMS is a computer-based system to support group meetings and the group work environment in general [Nunamaker 1991]. In practice, EMS is used to support decision making among group members who share common interests and goals. On the other hand, an NSS is an "interactive computer-based system intended to support adversarial negotiating parties and possibly a mediator in reaching an agreement" [Anson and Jelassi 1990]. Carmel et.al [1993] define NSS as " a system consisting of hardware, software, people, procedures, and data that assists the individual negotiator, negotiation team, and third party. The NSS provides solutions, advises, or facilitates the process of negotiation."

Various kinds of NSS have been developed for over a decade. However, "most of the existing NSS to date make suggestions to the users and they are used to support only a certain stage in the negotiation process" [Carmel et.al, 1993], and "none of these implementations are full-featured NSS" [Foroughi et.al 1995]. Furthermore, most of the existing NSS require sophisticated "Decision Room" setting [Carmel and Herniter 1989, Foroughi et.al 1995] which makes NSS less accessible.

In cases where face-to-face negotiation is not possible (for example the participants are separated geographically), or the participants are unable to meet face-to-face because they are too emotionally involved, NSS in a decision room setting is not suitable, but a NSS in a teleconferencing setting is useful to support these specific cases of negotiation.

In the last few years, more and more people have connected to the Internet, particularly because of the popularity of the World Wide Web. Many companies, large and small, have started doing business on the Web, and/or have created Intranets for internal business operations. The advent, availability, and popularity of multi-tasking operating systems for the personal computer, Client/Server technology, the Java language, and the Web, have all stimulated the possibility of developing a negotiation support system in a Web-based teleconferencing setting accessible anywhere around the world. The new technologies allow this system to support electronic communication which is architecturally neutral (open systems), requires no sophisticated system settings, and reduces the need for software installation.

The research which is described in this thesis has the following objectives:

(1) Developing a prototype of a Web-based CBSS, a negotiation support system which can be accessed on the World Wide Web.

(2) Evaluating the validity of CBSS as an alternative to face-to-face negotiations.

(3) Investigating the effectiveness and efficiency of CBSS as a negotiation support system.

This thesis examines the design, implementation, and testing of a prototype of a Web-based Collective Bargaining Support System (CBSS), which is a specific NSS in a "Teleconferencing" setting (Figure 1.1). This system was developed under the Windows 95(™) environment, with a Client/Server architecture, written in the Java programming language (Sun's Java version 1.0), and accessible through the World Wide Web. The resulting software can be used in settings of the Intranet, Extranet, or Internet.

The next chapter, Chapter 2, reviews some literature related to negotiation theory and existing NSS. Chapter 3 describes the technology foundation of the CBSS design. The design and implementation of CBSS are presented in Chapter 4. Chapter 5 presents the experiment, hypotheses, and data analysis of an experimental implementation of CBSS. Conclusions and future research directions are presented in Chapter 6.

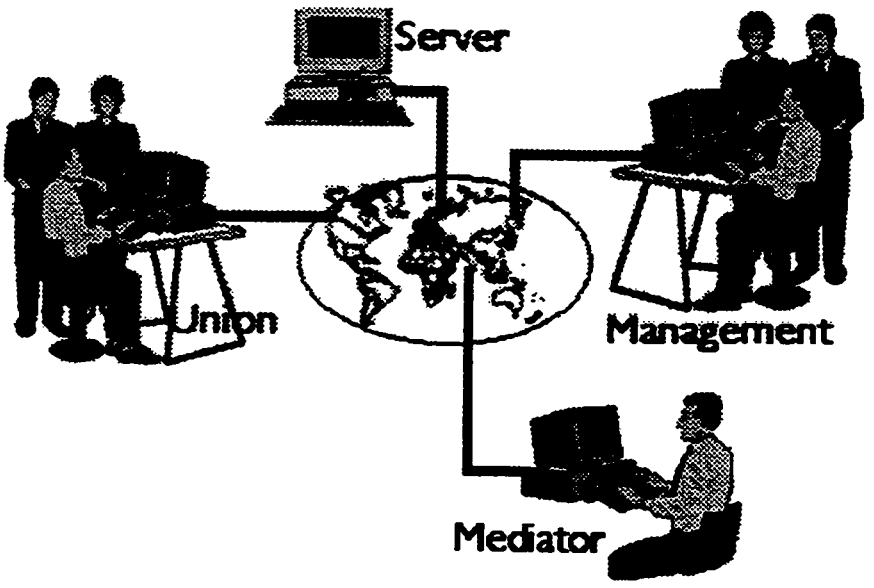


Figure 1.1 CBSS Teleconference Setting

Chapter 2. Literature Review.

2.1. Understanding Conflict

Conflict is a situation in which two or more motives, two or more strategies, or two or more objectives are mutually exclusive or partially blocking each other, and each of them satisfy only a portion of a given goal complex.

The views of conflict in organization have changed over time. In the classic management theory, the role of conflict was ignored. According to this classic theory, conflict in an organization could be eliminated, for example by redesigning the organization, because there should be no conflicts in a well designed organization [Rojot 1991]. However, regardless of organizational design, conflicts will be inevitable where human interaction occurs.

The original human relation theories considered conflict as an organizational disease to be cured. Conflict exists only because people misunderstood each other or because people have unsatisfied psychological and social needs. Conflicts could be avoided by designing the organization and job contents such that the psychological and social needs of the people were satisfied [Rojot 1991].

After the social turmoil of the 1960s, organizational theorists recognized that conflict was "...a naturally occurring organizational phenomena and that it had positive as well as negative consequences for different organizational actors" [Kochan and Verma, 1983].

Later organizational theorists such as industrial relations theorists suggested that conflict occurred because parties in organization have different preferences, goals and interests. The more heterogeneous the cultural, social, economic, and demographic characteristics of the participants in an organization, the greater the potential for conflict [Rojot, 1991]. They believed that conflicts were neither diseases nor results of faulty organization designs.

The modern view considers conflict to be an important aspect of group dynamics and recognizes that conflict needs to be managed properly rather than avoided. A lack of conflict among the decision makers can lead to poor decisions [Miranda 1994]. However, improperly managed or poorly managed conflict is also dysfunctional, since it inhibits effective problem solving, results in participants' dissatisfaction with the process and outcomes, and can lead to the disintegration of the group [Rahim 1985].

According to Deutsch [1969] there are two dimensions of conflict: "issue-based and interpersonal". Issue-based conflict is desirable because it focuses on task-related issues and helps the groups develop better solutions. Interpersonal conflict is undesirable because it targets persons within the group, detracts attention from the task, and damages the group's collaboration. Interpersonal conflict should be suppressed and issue-based conflict should be managed or resolved effectively. A system that assists in the resolution of conflict must assist

negotiators in reducing interpersonal conflict and support them in resolving issue-based conflict.

2.2. Negotiation Theory

The are many ways to resolve a conflict. For example: **combat** as in a fight or war, **chance** as in tossing a coin, **contest** as in a competitive exam, **voting** as in an election, **deferring to authority** as in court proceedings and **negotiation** as in business transactions [Raiffa 1982, Rojot 1991].

Negotiation is "a process whereby two or more parties attempt to settle what each shall give and take, or perform and receive, in a transaction between them" [Rojot 1991]. When parties are in conflict, they or their representatives conduct negotiations in an attempt to reach a mutually acceptable agreement that can resolve their differences.

Kochan and Verma [1983] proposed five basic points concerning the nature of conflict and negotiation in an organization:

> "(1). Organizations are inherently mixed motive in nature. Participants share some common interests and have some conflicting ones as well. Negotiation on conflicting interests is important because the parties share enough interdependence to stay together rather than terminate their relationship.
>
> (2). The differences that create conflict situations in an organization can vary considerably from objective differences to highly subjective, interpersonal differences. Negotiation strategies should be able to cope with these kinds of differences, manage the objective differences and handle subjective differences effectively.

(3). Power should be considered as an important source of influence. The parties should have some power to influence each other. The absence of shared power can lead to unilateral decisions. The more unequal the distribution of power the higher probability of a unilateral rather than a negotiated outcome.

(4). The lack of effective conflict management is likely to lead to lower levels of goal attainment for all parties. Effective conflict management requires that the resolution process be able to deal with the underlying sources of conflicts.

(5). Assessment of the negotiation's result or other organizational process needs to be made in terms of the extent to which they contribute to the goals of each parties involved."

According to Jelassi and Foroughi [1989]: "..negotiation situations can be seen as lying on a continuum from **hard** to **soft**". Hard negotiation is distributive bargaining, a fixed pie model, a win-lose proposition. Soft negotiation is an integrative bargaining, win-win proposition. In distributive negotiation, one single issue is under contention and the parties have almost strictly opposing interests on that issue. The more one party gets, the less the other party gets. If both parties are very greedy, the negotiation process may fail to come to an agreement. In integrative negotiation, the parties are willing to reconcile their interests. The parties communicate their interests and alternative solutions to the problem. They discuss the best outcome that presents high joint benefits. Integrative negotiation is important for resolving conflict in an organization because:

"- integrative agreements are likely to be more stable. Compromises (in distributive negotiation) are often unsatisfactory, causing the issue to come up again at a later time,

- integrative agreements are mutually rewarding and tend to strengthen the relationship between the parties,

-distributive bargaining may fail, so the parties should reconcile their interests in order to find a way to resolve their conflicts, and

- integrative agreements may contribute to the welfare of the broader community of which the conflicting parties are members." [Pruitt 1983].

Furthermore, Pruitt [1983] proposed five principal methods for achieving integrative agreements:

"(1) *Expanding the pie*. Some conflicts hinge on a resource shortage. An integrative agreement can be devised by increasing the available resources.

(2) *Nonspecific compensation*. The resistant party can be awarded valuable things from some unrelated sources.

(3) *Logrolling*. Each party concedes on low priority issues in exchange for concessions on issues of higher priority. It is useful to have information about each party's priorities so that exchangeable concessions can be identified.

(4) *Cost cutting*. One party gets what he or she needs while the other party's costs are reduced or eliminated.

(5) *Bridging*. Neither party gets its initial demands, but a new option is devised that satisfies the most important interest underlying those demands."

Kessler [1978] developed a structured, four-stage process model called Creative Conflict Resolution (CCR) to carry out the integrative negotiation process. Anson and Jelassi [1990] adopted the CCR model and divided negotiations into four stages: (1) setting the stage; (2) formulating the problem; (3) processing the issues, and (4) resolving the issues. Table 2.1 outlines their suggested negotiation process.

Table 2.1: Integrative negotiation process stages and activities (Anson and Jelassi, 1990)

Process Stages	Activities
Pre-Session	(A) Pre-negotiation strategy formulation (B) Agreement to engage in negotiation
(1) Setting the stage	(A) Establishing rules (B) Developing positive frames
(2) Formulating the problem	(A) Defining the problem (B) Defining the issues
(3) Processing the issues	(A) Tracking time deadlines (B) Focussing on specific issues (C) Role reversal (D) Paraphrasing (E) Maintaining equality
(4) Resolving the issues	(A) Generating alternatives (B) Analyzing alternatives (C) Evaluating issues (D) Developing solutions (E) Completing the agreement

Gulliver [1979] proposed an eight-step process model of negotiation. This model describes the developmental progress of the negotiation from start to end. These steps are outlined as follows:

> "(1) Find the Arena : parties discuss and agree on the location where the negotiation will take place;
> (2) Set the Agenda : parties discuss and agree on the issues to be discuss, the order of discussions, and the trades between issues if any;
> (3) Explore the Field: parties try to establish the limits to the issues in dispute;
> (4) Narrow the Difference: parties conduct the bargaining, look for possible solutions, and try to narrow their differences;
> (5) Preliminaries to Final Bargaining: parties refine persisting differences, test trading possibilities, search for a viable solution, and construct a bargaining formula if necessary;
> (6) Final Bargaining: parties exchange specific and substantive proposals and counter-proposals;

(7) Ritual affirmation: parties sign the agreement;
(8) Execute the agreement." [Gulliver, 1979]

Fisher and Ury [1981] proposed fundamental guidelines for conducting negotiations:

"(1) *Separate the people from the problem.* Negotiators must separate the people from the problem in order to reduce the negative impacts of misperception and emotion.

(2) *Identify the parties' real interests.* Negotiators often confuse their position on an issue with their fundamental interests. Parties do not correctly assess their own goals, objectives, and values; if they did, the parties in a negotiation might find that some of their interests coincide.

(3) *Generate options for mutual gains.* Negotiators are often blind to creative solutions because they assume that all possible solutions involve splitting the pie. Negotiators should focus on mutual gains, consciously trying to overcome the fixed pie assumption and they should try to make the process of choosing among options as easy and painless as possible for the other side.

(4) *Use objective criteria.* Negotiators need a common measure, or evaluation criterion. Focusing on fair standards and fair procedures is one way of developing objective criteria. Comparing contracts and economic circumstances within an industry and across different economic sectors is another way."

2.3. Collective Bargaining Theories.

Collective Bargaining is a process that usually involves dealings between representatives of the employees and employer, or between Union and Management, rather than between individual employees and the company manager. Collective Bargaining is a multi-dimensional concept. The five most significant dimensions are:

"(1) *Collective Action:* employees acting collectively to protect and improve their conditions of employment.

(2) *Recognition:* the requirement that collective action be supplemented by recognition from other individuals, groups, or organizations, in order to permit meaningful dialogue between the relevant parties.

(3) *Written Agreement:* the agreement between the relevant parties is a formal written contract that defines the agreement sets during the negotiation process.

(4) *Adherence:* the relevant parties will adhere to the agreement for a negotiated time period.

(5) *Deadline Procedures*: to be effective, a collective bargaining relationship must, at a specified point in time, allow the parties to invoke some types of actions designed to encourage the opposition to seek a meaningful settlement. Otherwise, it could be in the interest of one party to continue the negotiations for an indefinite time period." [Phillips 1977]

Walton and McKersie [1965] proposed a negotiation model for labor-management contract talks. They identified four subprocesses in a collective bargaining negotiation:

(1) *Intraorganizational bargaining.* This refers to the activities that are required to achieve consensus within a particular organization. It is a subprocess to bring the expectations of each party's members into alignment with their chief negotiator.

(2) *Attitudinal structuring:* This subprocess is designed to change or influence the attitudes and relationships which surround the negotiating parties. The parties develop friendly situations, create mutual trust, respect and cooperation.

(3) *Integrative bargaining*: This includes activities where there is a perceived area of common concern and the gains of one party do not represent equal sacrifices by the other. Thus, both parties can proceed to increase the size of the joint gain that is available to them.

(4) *Distributive bargaining*: This refers to a pure conflict subprocess in which one party's gain is another party's loss.

Prior to the Walton and McKersie [1965] analysis, distributive bargaining activities were frequently considered to be the only aspects of bargaining worthy of consideration.

The idea of distributive bargaining is most familiar in the form of a single-issue negotiation, such as bargaining for a used car; the issues or the problems are assumed to be areas in which objectives are in strict conflict. The common result of distributive bargaining is not always the best solution. In contrast, in integrative bargaining the problems are assumed to be areas of common concern in which objectives are not in fundamental conflict. The integrative bargaining approach attempts to steer the parties toward problem solving and away from the traditional zero-sum horse trading. There is a consensus that negotiations, particularly union-management contract negotiations, need to move in the direction of integrative bargaining [Cramel et.al 1993].

According to Gershenfeld, McKersie, and Walton [1995], collective bargaining is a negotiation over change involving both initiating and responding parties. Sometimes parties negotiate to revise the existing contract or agreement, or they negotiate to change their relationship in the organization. Why negotiate over change? Because acting unilaterally will be prompted by other party responds and at this point negotiations will be demanded. In many occasion parties act on the basis of consensus, but the consensus is rarely complete. Differences usually emerge and at this point negotiations will be needed. Both unilateral action and consensus will cause a back-and-forth process. This process may involve dialogue

and understanding in order to reach joint agreement, and the parties involved in the process are on the path of negotiation over change. Gershenfeld et.al [1995] discussed and presented case studies concerning two primary negotiation strategies: **forcing** and **fostering**, and the combinations of these strategies. These strategies are summarized as follows :

> "In the **forcing** strategy one side has the power to compel acceptance of the demands while the other side receives the demands because other alternatives could be more costly. This approach associates some degree of risks. Either the relationship will decline or the change may not be forthcoming. In the **fostering** strategy both sides cooperate in finding solutions to their common problems that leave them in better situation. This approach involves less risk, often takes considerably more time, and for an extended period may not show any benefits." [Gershenfeld et.al 1995]

The Forcing strategy in some ways is similar to the distributive bargaining approach while the fostering strategy is similar to the integrative bargaining approach.

2.4. Negotiation Support System.

In most business organizations, a decision making process involves choices among alternatives and is conducted by groups of executives in a face-to-face meeting. The assessment of risks and selection of a solution is a shared process. Gray and Nunamaker [1989] defined the main characteristics of a group meeting as follows: "the meeting is a joint activity, engaged in by a group of people of equal or near equal status, it is an intellectual activity, the product depends in an essential way on the knowledge, opinions and judgement of its participants, and the differences in opinion are settled either by fiat or by negotiation".

A Group Decision Support System (GDSS) is a computer-based system for

supporting the group decision making process. According to DeSanctis and Gallupe [1989], a GDSS is "an interactive, computer-based system which facilitates solution of unstructured problems by a set of decision makers working together as a group". A GDSS should improve the decision making process and decision outcomes of decision maker groups. A GDSS should encourage the group meeting participants to be active in the process and it should contain a mechanism to discourage development of negative group behaviours such as miscommunication, groupthinking and destructive interpersonal conflict.

A dynamic group decision making process almost always involves some levels of conflict. A special GDSS for managing these conflicts is called a Negotiation Support System (NSS). Jelassi and Foroughi [1989] offered a definition of a NSS as a special type of GDSS intended to support negotiation parties in reaching an agreement. Carmel et.al [1993] proposed a more comprehensive definition: "a Negotiation Support System is a system consisting of hardware, software, people, procedures, and data that assist the individual negotiator, negotiation team, and third party. The NSS advises, provides a solution, or facilitates the process of negotiation". Here, Carmel et.al [1993] differentiated NSS into two types, the solution driven NSS and the process support NSS.

The solution driven NSS provides solution alternatives or suggests agreement to the negotiation parties. The suggestions are made using a number of possible models such as: Social Judgment Theory Models, Hypergame Decision Models, Bargaining Models, Multiobjective Linear Programming, and Expert Systems [Carmel et.al 1993].

The process support NSS does not provide any quantitative solutions. It is designed to support the process of negotiation from preparation stage to contract signing stage. Process support NSS address two dimensions that the solution-driven NSS does not: enriched communication channels and cooperative work [Carmel et.al 1993]. The process support type NSS may be used either in a face-to-face negotiation (Decision Room setting) or in a "distributed-synchronous" [Burke and Chidambaram 1995] (teleconferencing setting) negotiation. According to Burke et.al [1995] a distributed-synchronous meeting (different places at the same time) is somewhat less interactive and permits fewer types of message transmission. However, according to Short et.al [1976] a distributed-synchronous meeting exhibits less social presence, is less expressive, less emotional, and is more de-personalized and more businesslike.

Lim and Benbasat argued that "it is important to distinguish between the effects of the decision support component of a NSS and its support for communication among participants" [Lim and Benbasat 1993]. The decision support component, which provides alternative solutions, affects negotiation outcomes such as the distance from the efficient frontier and the distance from the Nash solution. The communication support component would decrease the time to settlement and increase satisfaction with the results by increasing the commitment of the participants to a mutually satisfactory outcome.

Various kinds of NSS have been used for over a decade. However according to Carmel et.al [1993], most NSS systems are solution-driven. Most of these systems do not support interactive negotiation and they are used to support only a certain stage in the

negotiation process. For example, among eight NSS discussed by Anson and Jelassi [1990] only two support some form of multiple participant system input interaction, while the remaining systems are operated by a single negotiator. These systems support the process in a certain stage only, such as: pre-negotiation strategy formulation (Computer Decision Tree [Winter, 1985], Decision Maker [Fraser and Hipel, 1981], Rune [Kersten et.al, 1986]), and evaluation of alternatives (Decision Analysis [Executive Software Inc., 1983], DTG-Analytical Mediation [Mumpower et.al, 1986], Mediator [Jarke et.al, 1987], Nego [Kersten, 1985], Policy PC [Executive Decision Services, 1983]).

According to Anson and Jelassi [1990], the primary shortcomings of existing NSS are their lack of support for:

(a) handling cognitive bias and socio-emotional aspect of conflict situations,

(b) structuring the negotiation process,

(c) facilitating direct interaction between the negotiation parties.

Recently, a special type of GDSS, the Electronic Meeting System (EMS) has been used as a NSS (at least in a case study). However, according to Carmel et.al [1993], EMS does not address an implication of the negotiation task, which is the idea of "dyad" (two adversarial groups meeting together), because EMS is designed for a group meeting (group members share common interests and goals) and not for negotiation parties (parties have different interests and goals). The EMS can be adapted to the NSS form by adding software tools to support the dyadic approach and to facilitate process structuring.

Anson and Jelassi [1990] developed a framework for designing a type of process support NSS. They believe that NSS should address analytical processing, socio-emotional and cognitive obstacles, and it should assist the negotiators' interactions or communications during all stages of the negotiation process. Several general characteristics of NSS have been advocated by many researchers [Anson and Jelassi 1990, Carmel and Herniter 1989, Fisher abd Ury 1981, Jelassi and Foroughi 1989]:

(1) improve communications among the participants,

(2) generate alternatives before judging,

(3) separate personalities from the problems,

(4) use objective data and criteria.

2.5 Existing NSS Software

NSS has been around for some time and various kinds of NSS software have been developed and used successfully in a variety of settings. Some of these approaches are reviewed in the following paragraphs, including: CAP, DECISION MAKER, NEGO, DECISION CONFERENCING, MEDIATOR, RUNE and MEDIANSS.

2.5.1 CAP (Conflict Analysis Program). [Fraser and Hipel 1984]

CAP is an interactive, microcomputer based system designed to support a third-party

arbitrator at the pre-negotiation (strategy formulation) stage. Information representing the conflict situation is formulated into a game. CAP uses metagame analysis methods to formulate and analyze subjective alternative strategies and strategy preferences. CAP provides procedures to remove infeasible alternatives, to rank alternatives according to player preferences, and to analyze the stability of alternatives. However, this system neither supports group communications nor negotiation process structuring. It is a solution-driven NSS to support a single negotiator.

2.5.2 DECISION MAKER [Fraser 1993]

DECISION MAKER is an enhanced version of CAP. It is available both in DOS or WINDOWS environments. This software is useful for modelling and analyzing situations involving strategic negotiations. The solution space is all binary combinations of player alternatives. DECISION MAKER forecasts possible compromise resolutions and optimizes decision making, seeking stability for all participating parties. It is an iterative process that involves updating previous models after interpreting the stability of the obtained results. This software supports neither face-to-face negotiation nor bargaining processes. According to Fraser [1993], DECISION MAKER has been used by many people successfully. However it is a solution-driven NSS which does not support group communications, process structuring, and discussion documentation.

2.5.3 NEGO [Kersten 1985]

NEGO is a NSS mainframe software written for the IBM 370/148 under a VM/VSP environment. This software uses a two-stage interactive process leading to a compromise solution based on Kersten and Szapiro's generalized theory of negotiation formulation. In stage 1, the problem is formulated for each participant. In stage 2 these formulations are analyzed or negotiated. NEGO provides an iterative procedure to allow the participants to change their strategies, to form coalitions, and to compromise on the issues. The conflict situation is formulated into a multi-objective linear programming problem. There is no computerized communication facility provided, and all participants must remain in the same room, so that they can interact personally with each other. NEGO is a solution-driven NSS which does not provide group communications, process structuring, and discussion documentation. NEGO had been used in at least five management courses with some case negotiation case studies.

2.5.4 DECISION CONFERENCING [Jelassi and Foroughi 1989]

This software provides support for a participant in the pre-negotiation stage. It provides structure for the dis-aggregation of complex issues, anticipates the positions of other participants, and creates new alternatives. For use in direct negotiation, decision models are developed separately for the two opposing parties. These parties can work together with a model to derive a mutually preferred solution. This software combines decision analysis

theory with group processing techniques drawn from organizational development. It is designed for a computer-based conferencing room environment. This system is a solution-driven NSS type which does not provide discussion documentation and process structuring.

2.5.5 MEDIATOR [Jarke, Jelassi and Shakun 1987]

MEDIATOR was designed to support a multi-agent decision making environment involving conflicts and was based on a database-centre model. During the pre-negotiation stage, this software allows the participants to formulate their initial bargaining position. In the negotiation stage, this software helps the participants to select and to evaluate the alternatives. Each participant is connected to a DSS which is a single user multicriteria DSS called PREFCALC, to establish preferences. Participants and a mediator share a common database. The human mediator integrates these individual problem representations using relational query language capabilities to form an initial group joint problem representation. Negotiation occurs by consensus seeking through exchange of information and compromise. MEDIATOR can be categorized as a solution-driven NSS which provides support for individual negotiators in the preliminary stages and group communications in later stages. The software was tested in a group car buying decision process during the prototyping process.

2.5.6 RUNE [Kersten et.al 1986]

RUNE is designed as an AI approach to NSS. It is a rule-based expert system to help

the participants model and analyze their strategies. It views the negotiation as a two-stage process, the learning stage and interaction stage. In the first stage the participants formulate their initial proposals, while in the second stage, they exchange proposals and make concessions, reaching either a compromise solution or a deadlock. RUNE support facilities include: (1) a set of tools that analyze the alternatives and check for inconsistency, (2) a goal modifier that updates the content of the rule bases, and (3) an inference engine that carries out deductions at the meta-rule level. It is a solution-driven NSS which provides some form of group interactions. No real negotiations using RUNE had been reported yet.

2.5.7 MEDIANSS [Carmel and Herniter 1989]

MEDIANSS (MEDIAted Negotiation Session Support) is an experimental software system at University of Arizona, and is implemented as a part of the PLEXSYS GDSS in a specific "Decision Room" setting. It is designed for a multi-user environment to support negotiation sessions. The participants are guided by a structured set of computerized steps by the mediator. The target situation for this system is a two-sided dispute with each side represented by teams. The main task of MEDIANSS is to focus on the negotiation session. The structured steps consists of 10 steps: Rule setting, Role reversal, Issue and reason identification, Issue consolidation, Ranking, Create package, Present proposal, Linking, Horse trading and Agreement wording. This system is an example of a process support system for the negotiation process, but an organization wishing to use it must also install this system in a setting similar to the PLEXSYS GDSS of University of Arizona.

2.5.8 EMS as NSS [Carmel et.al 1993, Nunamaker et.al 1991]

Dennis et.al [1988] defined EMS as "An information technology-based environment that supports group meetings, which may be distributed geographically and temporally...". EMS are now being augmented with a negotiation-specific module to provide support for negotiations [Jelassi and Foroughi 1989]. Nunamaker et.al [1991] and Carmel et.al [1993] had conducted a research study on the usage of EMS as an NSS. According to Nunamaker et.al [1991] EMS can enhance the initial phase of negotiation, and improve the generation of options for mutual gain. According to Carmel et.al [1993] bargaining sessions broke the EMS mold because it was impossible to pretend that the two sides were really one, yet contract talks require that the system recognize two groups and maintain the confidentiality of at least some of the communications and some of the data.

2.5.9 An Experimental NSS [Foroughi, Perkins, and Jelassi 1995]

Foroughi, Perkins, and Jelassi [1995] designed an experimental NSS in order to investigate the effects of a process support type NSS on negotiation time, negotiator satisfaction, and joint outcomes of certain conflict situations. This NSS was designed to support an integrative bargaining process in five stages: (1) statement of interests, (2) role reversal, (3) searching for common ground, (4) generation and analysis of alternative solutions, and (5) reaching agreement. The experimentation took place in the Collaborative Work Support Laboratory at Indiana University. Two software tools were used: the Topic

Commenter of the University of Arizona GDSS system, and the Negotiation Decision Support Tool (NDST). This system was implemented in a "Decision Room" setting. Probably MEDIANSS and this experimental system can be categorized as the first complete process support NSS. However, both system were implemented by using University of Arizona GDSS system which required a specific setting. So, any organization wishing to use this facility must install the similar system or ask its negotiation teams to go to Arizona.

2.5.10 INSS [http://www.business.carleton.ca/interneg]

A Web-based negotiation support system named INSS (Internet Negotiation Support System) was developed at the School of Business, Carleton University, Canada. The system was implemented by using Java Script language. This system can be categorized as a solution-driven NSS which provides a method to construct negotiators' utility function (based on issues rating and their options rating) to evaluate other side proposals. INSS also provides suggestions or solution alternatives to both sides by comparing both negotiator's utility functions.

INSS is a very good system for negotiating issues which can be integrated into one package, such as in a car buyer-seller negotiation, where price, warranty, and model issues can be integrated as one package. INSS however does not allow the negotiators to negotiate an issue separately from other issues.

2.5.11 Collaboration System (Groupware)

There are several commercial software packages available to support group processes, such as Lotus Notes, Microsoft Exchange, OracleMail, Collabra, etc. These software are categorized as Groupware to support computer-mediated collaboration process such as: electronic meeting, video conferencing, and group scheduling. The bottom line difference between the existing groupware and a Web-based NSS is design philosophy. Groupware is designed to support a group of users who are doing collaborative work, not two adversarial groups who are negotiating conflicted interests. Groupware can be categorized as EMS, it can be used as an NSS with some adjustments.

Most of the groupware are designed as pure internal applications (on a company's LAN), and client systems must be installed on each user terminal. Some Web-based groupware have been implemented, but installation of the client systems is still needed on each user terminal.

Web-based NSS can be used as an internal and external application (from anywhere around the World), and require no client system installation. A system such as Lotus Notes was designed as a proprietary system in an era lacking widespread connectivity, where the data structures are replicated in each terminal. A Web-based NSS is designed to take advantage of the Internet's worldwide network and eliminate the need to replicate data structures.

2.6 Research Directions.

Various kinds of Negotiation Support Systems (NSSs) have been developed to date. However, some of these NSSs (CAP, DECION MAKER) neither support group communications nor structure the negotiation. They support single negotiators in the pre-negotiation stage to evaluate alternative solutions. Other NSSs (NEGO, DECISION CONFERENCING, MEDIATOR, RUNE, INSS) support some forms of group communication but only on solution models or solution alternatives. There are some NSSs (MEDIANSS, EMS as NSS, Experimental NSS) that provide process support (group communication, process structuring, documentation) but they require a specific decision room setting. The complexity and the cost of implementing decision room settings is probably the main reason why these system are not widely used, especially for some organizations which use such facilities infrequently. The Negotiation Support System designed in this research is a process support NSS which requires no sophisticated setting, inexpensive, and easy to implement. It therefore has a potential for wider acceptance.

The objectives of this research are: 1) to build a prototype of a Web-based Collective Bargaining Support System (CBSS), a process-support NSS which can be accessed on the Web, to support the Collective Bargaining process. 2) to evaluate the validity of CBSS to be used as an alternative of face-to-face negotiations, and 3) to investigate the effectiveness and efficiency of this system as a negotiation support system. The characteristics that differentiate this system from existing NSS are:

(1) The system supports process where two parties are in adversarial positions. It is different from EMS or other groupware support which are for meetings with non-adversarial participants.

(2) The system facilitates the negotiation process between two parties and also can support a third party as a mediator. It is different from solution driven NSS which provide advice and alternative solutions.

(3) The system runs on a Web platform. This allows users to work at a Web client terminal anywhere in the world (teleconferencing setting), with their existing systems without any complex setting. It is different from existing NSS which require a specific network setting such as "an electronic meeting room".

(4) The Java language is used to develop the system, which guarantees platform independence and system robustness. It is different from existing NSS which are mostly platform-dependent.

The CBSS prototype is a text-based NSS, which supports the process of negotiation by providing electronic communications, online documentation, and process structuring. CBSS can be extended in the future by adding modules such as: fuzzy negotiation model, audio, video conferencing, and specific decision support system modules.

Chapter 3 : Technology Foundations of Collective Bargaining Support System (CBSS).

The first objective of this research is to develop a prototype of a Web-based Collective Bargaining Support System (CBSS), a process-support type Negotiation Support System which can be accessed on the Web. To achieve this objective, the system design was based on the most advanced information technologies such as: client/server systems, the Java language, and the World Wide Web.

The first version of CBSS was developed by using C^{++} language on personal computers (PCs) with serial cable connection. The CBSS system must be installed on each terminal to be used as a negotiation support system. It is a traditional approach. The system worked well, but it has some disadvantages. In order to use this system as an NSS, a specific setting with several PC terminals connected with serial cables must be established. Serial cable connection has a distance limitation which makes remote negotiations are not possible. During the early stage of CBSS development, the new technology such as Java language was not available yet, and the World Wide Web platform was not widely used.

Development of the second version of CBSS was started during the fall of 1995. Java language was just announced and distributed free of charge by Sun Microsystem. This version was implemented on personal computers connected by a local area network (LAN) supported

by TCP/IP protocol. The CBSS programs was converted from C++ to Java language. The second version allowed remote negotiations as long as computer terminals were connected with the company's LAN. However, some problems still exist, negotiations by using this system from terminals which were not connected by a LAN were not possible, and the system installation was still required on each terminal. Each time the system program was modified, even just a little modification, the system must be installed on all terminals.

In the third version of CBSS all disadvantages of the previous systems were eliminated. The only way to overcome the previous difficulties is by using the World Wide Web as the networking platform, the Java as the programming language, and Client/Server as the computing architecture. In this chapter, we will discuss the technology foundation related to the development of CBSS.

Client/Server (C/S) computing has attracted the attention of many organizations because of benefits that have been reported in many journals and magazines, such as: cost savings, productivity increases, flexibility, and better resource utilization. For example, Sunoco Inc. reported that the cost of computing was reduced by 20 to 30% after converting its existing centralized computing system to a C/S computing system [Zeidenberg, 1996]. Client/Server architecture splits the workload between client computers that request services and server computers that process the request. C/S takes advantage of the computing power of desktop computers by letting them share some of the processing burden. It is a modular information system architecture that makes information more accessible, which in turn results in better customer services, tighter inventory control, and a more effective sales force. Client/Server systems make data more accessible to users, since any legitimate clients

connected to a server machine can access the data, and manipulate and display the data by using their own software. C/S systems also provide a modular architecture that makes it easy to build flexible end-user applications, and move away from proprietary solutions to more open solutions. C/S computing is a computing paradigm that can be used to replace host-based or mainframe computing systems (centralized computing).

Internet, the network of networks, is essentially a huge C/S system. The servers, which are located at many sites, provide clients with access to resources and information the clients do not have on their own host or server system. The Internet has gained more popularity through the advent of the World Wide Web (WWW) or Web. The Web has become the most popular networking platform these days because it provides the technology to enrich the content of information provided for the users (it includes all multimedia forms with hyperlinks), and it can be accessed from various platforms around the world. The Web uses a C/S architecture for distributed hypertext that can be accessed over the Internet. Servers provide data to clients in a standardized hypertext markup language (HTML) through a standard hypertext transfer protocol (HTTP) such that users can access information from any servers, from any places, and from any systems connected to Internet.

Java, the new programming language developed and introduced by Sun Microsystems, has many characteristics that meet the requirements of the Web as an open system. The language is architecturally neutral and it has the potential to be the language of multimedia system programming. The well known potential of using Java includes transforming a static Web page into a dynamic Web page with animation and real interactivity, and the potential to deliver software across multiple platforms.

The strengths and weaknesses of these three technologies are summarized in the following Table 3.1. A Web C/S systems gains many advantages over traditional C/S systems by using the Web as its networking platform and Java as its programming language. The most obvious advantage is that the designer needs to write an application only once to be distributed to clients on the Internet regardless of the client hardware. The problems of software installation, software upgrading, and multi-platform software development are solved.

Table 3.1 : Strengths and Weaknesses of C/S, WWW, and JAVA

Technology	Strengths	Weaknesses
Client/Server	- long run cost reduction - increase efficiency of the existing equipment - increase productivity - flexibility	- desktop centric / architectural dependent - complex and difficult in development, installation, and maintenance
World Wide Web	- open system, platform free - multimedia contents - accessible around the world	- security - application conversion into Web standard - static content - stateless domain - limited database access
JAVA	- simple, robust, portable, and general purpose - architecturally neutral - software distribution on time - software installation and software de-installation performed automatically - network centric - dynamic and multimedia support - reduce development, installation, and maintenance problems.	- much slower than C++ - still in a third generation language format - new technology, not mature

3.1. Client/Server System

A Client/Server system can be described as: "a computational architecture that

involves client processes requesting services from a server process connected by middle ware" [Dewire, 1993]. The three main components of a C/S system are: the **client**, the **server**, and the **middleware**. At the client side there is a program that sends messages to the server process through middleware to ask for services. A program at the server side fulfills these requests by performing some tasks, and sends the results back to the client through the middleware. The middleware connects the client and the server by providing a communication channel, scheduling, security, transaction management, and so on. The middleware may be located on the server machine (such as a Remote Procedure Call), on the client machine (such as a Microsoft's Open Data Base Connectivity driver), or in the middle (the network), depends on the type of services, and the type software and hardware used.

Client/Server computing is a logical extension of modular programming [Chorafas, 1994]. The basic assumption of modular programming is that the separation of a large piece of software into several modules will make the development and maintenance of the software easier. Client/Server computing takes a step further by recognizing that these modules need not all be executed within the same memory space; they may be executed on different machines and at different locations. With this architecture, the modules may be classified into two main classes, the calling modules and the called modules. The calling module becomes the "client" (that which requests a service), and the called module becomes the "server" (that which provides the service).

Each module in Client/Server computing can be run on a different platform, under a different operating system and with a different network protocol. Each module can be developed and maintained separately. Data can be placed closer to the user. Users can access

their data through a user-friendly interface using tools for manipulating their data into meaningful information. This allows users to do more for themselves, rely less on the ISD (Information System Department) for assistance, and receive better turnaround time for applications requested of IS. The end result is more efficient use of existing equipment, increased effectiveness of applications, and more productive workers. However, there are some problems with the usual C/S system (basically, because this system is platform dependent):

(1) *Installation problems*: the client software must be installed on the client machine. If there are a thousand clients, the software must be installed a thousand times.
(2) *Development problems*: if the C/S system involves multiple platforms, the client software must be developed many times, once for each platform.
(3) *Maintenance problems*: if the client software is upgraded, the replacement software must be installed on all client machines, so the replacement program must be developed specifically for each client and server platform.

3.1.1 C/S Evolution [Chorafas 1994, Schussel 1996]

The oldest type of C/S system is the File Server System. In this system, if the client requests some data from the server, the server transfers all files that contain these data to the client. It may happen that the whole database must be transferred in order to fulfill the client request, and the transfer of a large database will increase network traffic and slow the system response.

File Server Systems were eventually replaced by Database Server Systems. Whenever a client needs data, it sends a request to the server. The server accepts this request by running a database query and sends only the data requested. In this case, network traffic is reduced significantly.

An architecture known as **2-tier** was introduced later, where Structured Query Language (SQL) is used by the server to fulfill client requests. However, as the number of clients increases the performance of this C/S system deteriorates. Another problem is the difficulty in re-organizing business system logic from one server to another if a new application is developed.

The **3-tier** architecture is a more advanced approach which remedies limitations of the 2-tier architecture, by adding middleware to the system. The middleware performs various functions such as: queuing, scheduling, connecting, performing transactions, database staging and so forth. In this approach, the client and the server do not communicate directly to each other, but through the middleware. The client's request is sent to the middleware, the middleware schedules this request and then asks the server to fulfill this request. This middleware smooths and lowers the overhead of accessing the server. For example, a generic SQL translator such as Microsoft's Open Data Base Connectivity (ODBC) can be used as database middleware. The ODBC driver can be installed in the client's machine to translate database requests into a language understandable by the server. In another situation, middleware is a special Remote Procedure Call (RPC) stored on the server machine to handle any requests from the clients.

What is the next? According to Daniel [1996] ".... the next wave of C/S computing is an intergalactic era, an era spawned by exponential network growth and the arrival of network-aware multithreaded desktop operating systems, in which servers are plentiful instead of scarce, because every client can be a server, and in which proximity no longer matters..."

3.1.2 Hardware and Software Evolution.

Since computer hardware evolves every day, the hardware that you buy today will be obsolete in a short time. The memory capacity, speed, storage, and computer interface are changing rapidly. In less than two decades, the usual main memory of a PC has grown from 4 Kilobytes to 64 Megabytes, or 16000 times. The speed of the processor has increased from 8 MHz to 200 MHz, hard disk storage from 340 KB to 3 GB. The screen has changed from monochrome to Super VGA, the 2.4 Kilobit serial cable has been replaced by 10 Megabit Ethernet, and so forth.

The operating system for the PC has evolved from 8-bit to 32-bit memory addressing, from single threaded to multitasking. Recent operating systems include many features of the mainframe such as: virtual memory, multithreading, communication support, and Graphical User Interface (GUI) front-ends.

Networking systems among PCs have also been evolved rapidly. The earliest Local Area Networks (LANs) were used primarily to share devices (e.g., laser printers). The next step was to allow data sharing in addition to device sharing. These days the LAN has evolved

into connections through Metropolitan Area Networks (MAN), Wide area Networks (WAN), and the Internet.

The evolution of computer hardware and software has supported the evolution of Client/Server systems. Without the rapid advancement of computer technology the evolution of Client/Server systems would have been very slow, if they developed at all.

3.1.3 C/S Applications Categories.

C/S applications can be categorized into classes based on where most of the processing is done. Each class may require different hardware and software capabilities on the client, server, and the middle-ware [Dewire 1993]. The categories are [Dewire 1993]:

1. *Host-based processing* : all processing is done at the server machine.
2. *Client-based processing* : all application logic is run on the client machine.
3. *Cooperative processing* : divides the application load between the client and the server.

Another way to categorize the C/S system is by looking into what process the system supports [Dewire 1993].

1. *Office system extension* : file sharing, e-mail, ftp, group work, etc.
2. *Front-end* : legacy data system on the server is displayed as GUI products on clients.
3. *Database access* : client queries on the server's database.

4. *Transaction Processing applications* : order-entry, inventory, point-of-sale, etc.

5. *Investigative application* : Decision Support Systems, Executive Information Systems, etc.

CBSS was implemented as a cooperative Client/Server system. The server system provides services such as: messages distribution, file handlings client identification, and communication services. The client system provide user interfaces, data handling, download and upload process, and connection request.

3.2. C/S System and The Internet.

The Internet is composed of interconnected computer networks which communicate using a standard protocol, the Transmission Control Protocol/Internet Protocol (TCP/IP). The Internet has been rapidly popularized and commercialized, particularly by the World Wide Web (WWW or the Web) popularity. The availability of the Web as a widely spread platform operating across all relevant hardware platforms on the Internet has made the Internet the dominant networking platform today. Some experts speculate that the Internet's ubiquity and relative openness will wipe out the benefits of the C/S structure, but according to Mitchell Kertzman: "The Internet will enhance, rather than make obsolete C/S computing" [Picket 1996]. In addition, Internet's other applications (i.e. E-mail, Gopher, FTP, etc.) are essentially C/S applications. According to Daniel [1996]: "...with more and more organizations getting wired, the Web is emerging as C/S next big wake-up call...".

3.2.1 World Wide Web.

The World Wide Web (WWW), well known as the Web, is the most recent and the most powerful information service on the Internet. Web provides information with text, graphics, image, sound, and video contents, and it also integrates other services such as E-mail, FTP, News, etc. The interesting characteristic of the Web is that it is a platform independent C/S system. Web servers and client browsers can run on various platforms and interact with each other.

The Web project was started by Tim Berners-Lee at the European Particle Physics Laboratory (CERN) in Geneva, Switzerland. The main objective of this project was to find a way for scientists doing projects at CERN to collaborate with each other through the computer network. The project was started in March 1989, and the first product of this project appeared on the Internet in January 1992, using the Hypertext Transfer Protocol (HTTP). HTTP is a protocol for exchanging hypertext documents. Hypertext documents are documents written in Hypertext Markup Language (HTML) and they are called HTML files. Web servers store a variety of multi-media documents in HTML file format. Clients use a Web browser to read this HTML file and to display the information.

The Web has spawned the creation of many Web browsers. The most popular at the current time are Netscape Navigator and Microsoft Internet Explorer. These browsers form a machine-independent platform on which software developers can build client applications for the Web C/S system. Since the Web can be accessed world-wide, it may become a way to increase the level of reach of any business organization, and by using the browser as the

platform for the application, the development problems of C/S systems can be solved.

The first generation of Web applications was characterized as a new form of electronic publishing, or multimedia publishing. This technology is a very successful in disseminating information but it is a "stateless connection" [Schussel 1996]. That is, the server does not keep track of who is requesting information or what was the last request from that user. The other limitation is that there were no database management systems connected to Web Servers.

The second and current generation of Web applications bring a newer and more capable environment by providing interactivity between the client and server (i.e. forms submission). It supports a more dynamic environment, allows clients to download applets (Java programs for the Web), to make queries on certain databases, and provides live Data Base Management Systems (DBMS) that enable the server to identify a client from page to page and from visit to visit. The current limitation is that most DBMS engines on the Web support only text-based data types. There are no major DBMS products that support the more complex data types such as voice, video, and images [Schussel 1996]. DBMS vendors such as Oracle have begun to develop and market this kind of support, in the form of object-oriented databases.

By using the Web as the networking platform for a Client/Server system, a number of advantages are gained over the traditional Client/Server system, such as :

(1) *A single version of client software will run on all platforms.* In the traditional Client/Server system, the developers are forced to create client software for every platform

this software must run on. By using the Web as the platform, the developers need to write only one set of code. In this case, a version of CBSS system is put under a Web server, and it can be downloaded by any clients who access CBSS site.

(2) *Modularity*. Once an infrastructure is in place for access to the system via Web browsers, it becomes simple to make modifications to the system. The developers modify only one set of code. In this case, only the version of CBSS which is put under a Web server needs to be modified and all clients who access CBSS site will get a new version automatically.

(3) *Uniformity*. The Web browser becomes a standard access method on the Web, and user training becomes easier and less expensive, especially for users who are familiar with information retrieval on the Web. The transition of users between applications also become easier because the interface is uniform.

3.3 JAVA Language.

Java as a language started its life as a programming language for consumer electronics, in devices such as toasters, microwave ovens, personal digital assistance, and so on. A small group at Sun Microsystems, headed by James Gosling, conducted research to develop software for consumer electronics such that it would be hardware independent, and the software would not change when the chips in these devices were replaced by new generation chips. This group quickly discovered that existing programming languages such as C and C++ were not suitable for this task because they had to be recompiled every time new hardware is installed. In addition, the complexity of C and C++ makes it extremely difficult to develop reliable software.

As a result, in 1990 Gosling started designing a new language that would be more suitable for consumer electronics. This language was originally named Oak, inspired by a tree outside Gosling's office window. Later, Gosling found that Oak had been used already for a previous language, so a new name was invented. Inspiration struck Gosling on a trip to a local coffee shop, and the new language was named Java. On May 23, 1995 the Java Programming Environment was formally announced by Sun at SunWorld '95.

3.3.1 The Nature of Java

Sun defines Java as "A simple, object oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high performance, multithreaded, and dynamic language". These terms will be examined in the following:

Simple: Java is a simple language. Computer programmers can learn this language quickly, because it is similar to but simpler than C or C++. The designers of Java removed a number of complex features available in C and C++, which could lead to poor programming practice or were rarely used. The conversion of C++ version of CBSS into Java language was done quickly because of this similarity, and yet the Java version of CBSS was less in number of lines.

Object-Oriented: Java is a real object oriented language. Although C++ is considered to be an object-oriented language, it still allows programmers to write non object-oriented programs, as in other procedural languages such as Fortran, Cobol, and Basic. In Java,

everything must be defined as an object or class, consisting of data and methods. This requirement makes Java version of CBSS becomes more modular than the previous version.

Distributed: Java is a distributed language. Java is designed to support applications on networks. Java provides a group of classes for network operations, defined in a package called "java.net". Java allows applications to read remote files as easily as local files. This is a suitable choice for distributed client and server systems. By using Java language to develop CBSS, the CBSS system becomes a distributed system naturally.

Interpreted: Java is actually a compiled and interpreted language. Programs written in Java must be compiled into byte-code, rather than into native machine code. It is the byte-code of the Java program which are distributed over the network. To run the program on a specific machine, these byte-code are interpreted by the Java interpreter in the browser software or the client machine. The Java interpreter and run-time system collectively form the Java Virtual Machine (JVM). JVM makes the platform independency becomes possible.

Robust: Java is a robust language, and facilitates programming reliable software. Java is a strongly typed language, which allows for extensive type-mismatch checking during program compilations. Java does not allow programmers to manipulate computer memory directly, and eliminates support for pointers, thus eliminating the possibility of overwriting memory and corrupting data. Java provides automatic garbage collection which prevents memory leaks and other bugs related to dynamic memory allocation and deallocation in languages such as C++. In addition, Java has an exception-handling feature, which generates a signal if an exception has occurred.

Secure: Java is a secure language. Since it is to be used in a networked environment, security is an important consideration, as users would not want to download a Java applet from a remote site without some assurance of security. Automatic memory allocation is one way to prevent maliciously written code, and Java does not allow programmers to set pointers for memory and change the contents of memory. The Java compiler does not handle memory layout design, so programmers are unable to guess the actual memory layout. Actual memory layout is designed at the time of byte-code interpretation by the Java Virtual Machine. The JVM also has a mechanism to verify byte-code loaded over the network in order to avoid any code that violates Java language restrictions such as Web viruses. In this case, Java version of CBSS system which is distributed through the Web is secure or safe to use on clients system.

Architecture Neutral: As a language for network-based applications, it is important that Java applications can run on any client platform on the network. Java programs are compiled into byte-code rather than into machine-code, so compiled Java programs are machine independent. The JVM residing on the client machine translates the byte-code into machine-code. This is an important feature in order CBSS to be designed as an architecture neutral system.

Portable: Since Java applications can be run on any system with JVM, Java is suited for portable applications. Java ensures this portability by not allowing a particular machine to implement different definitions for fundamental data types such as integers or bytes. By executing applications in an interpreter environment, Java applications do not have to conform to any single hardware platform. Compiled Java applications can be run on UNIX,

Windows 95, Windows NT, and the PowerPC Macintosh operating system without recompilation.

High-performance: The Java language supports several high-performance features such as multithreading, just-in-time compiling, and native code usage. As an interpreted language, Java is slower than C++, but it is fast enough to run interactive applications.

Multithreaded: Java is a multithreaded language. Multithreaded applications are capable of executing multiple tasks at the same time. The multithreaded characteristic is important for Web applications, since for example, a user could be listening to music while scrolling a page, while in the background the browser is downloading a video clip. This is an important feature to fit in CBSS development since CBSS system requires multithreaded tasks, such as receiving and displaying incoming messages while downloading user's notes from the server.

Dynamic: Java is a dynamic language, and is designed to adapt to an evolving environment. Java loads in classes as they are needed and the run-time class definitions in Java make it possible to dynamically link classes into an executing system.

The elements of the Java language are presented in Appendix B.

3.4. JAVA, WEB, and C/S Systems.

To summarize the previous sections, Java has gained popularity in the Web environment for its capability to create real time interactivity for Web users, and its capability to deliver Java applications (known as Java applets) on time. The other exciting part of Java is that Java applications are completely portable, programmers write and compile Java programs only once, and these applications can be executed on many machines that support Java. The Java compiler does not produce machine specific instructions, but an object in byte-code format. The Java Virtual Machine installed on the user's machine as part of its underlying operating system or as part of its Web browser will translate this byte-coded object into actual machine specific instructions. The Java Virtual Machine (JVM) is designed to check the byte-code object for invalid access to local system resources (memory, file system, etc.). Networking packages released with Java can be configured according to local security requirements. In this matter, Java codes are more secure than other programming language codes.

The massive growth of the Web demands a portable and distributed programming platform in which application can be embedded. The Java programming platform can match these requirements. Since its release in May 1995, Java has become heavily used on the Web. Some reasons that conform why Java is suitable for Web applications are as follows [Paolo et.al]:

(1) Java conforms with the Open System approach of the Web

(2) Java offers software distribution on-time easily by applets embedded in Web pages.

(3) Java provides run-time linking to simplify the transport of software agents.

(4) Java solves some Web security problems.

Most C/S applications today are developed for the desktop-centric model of computing. In other words, applications are designed with a specific client-architecture in mind (e.g. Mac, OS/2, Warp, Unix, or Windows). These desktop-centric applications must be pre-installed onto client PCs. If there is a change in the application the upgraded applications must be installed on a client-by-client basis. The integration of Web and Java technology changes the situation from desktop-centric computing into network-centric computing. Java applets which are embedded in an HTML file are distributed automatically to the client machine whenever the page is accessed, and no software installations, nor software upgrading, are required on the client machine. The following figure (Figure 3.1) shows how Java applet distribution takes place across the Internet.

Whenever a client accesses a URL through a Web browser, the designated Web server sends the HTML files to the client. If the HTML file contains an applet tag the server will send a Java applet immediately to the client machine. The Java Virtual Machine (JVM) translates the applet into client machine code and executes the application.

Java creates a new computing paradigm in the C/S system, including:

- Java allows the developer to create applications that eliminate much of the communication traffic between the client and the server.
- Java applications reduce server congestion, since they run on the client machine.

- Application installation and de-installation are performed automatically.
- Java eliminates the concept of application upgrading, since upgrading is done only on the server side.
- Out-of-date software versions are eliminated, since the clients always get the latest version.
- Creating and maintaining different platform versions of software are eliminated.

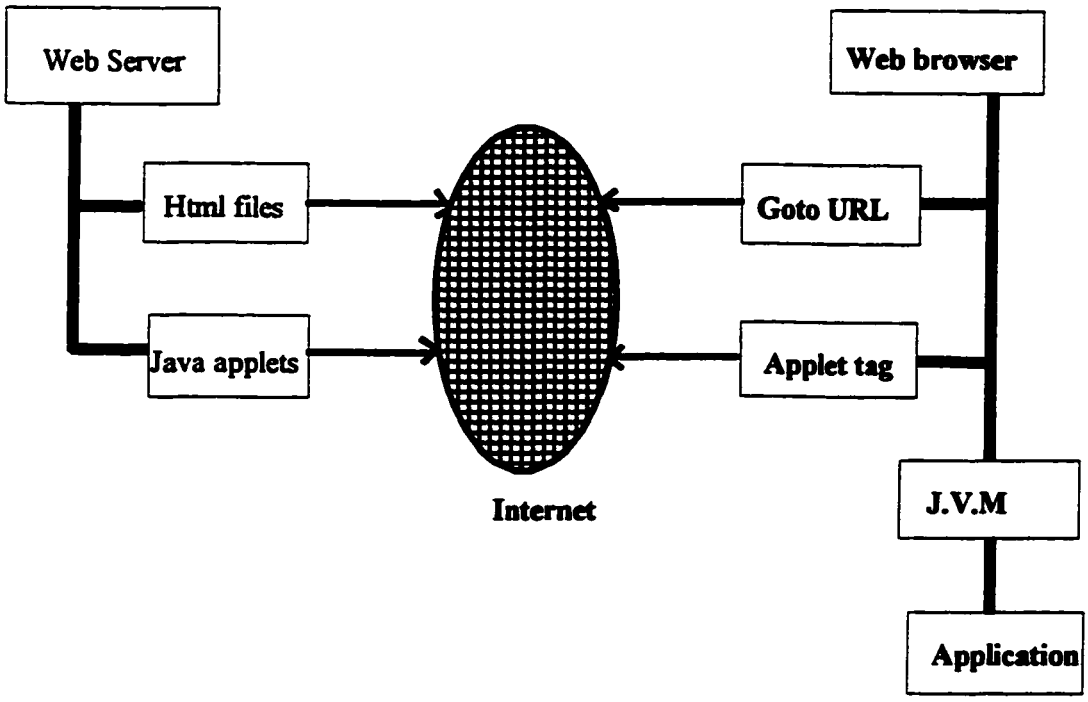


Figure 3.1 Distribution of Java application.

Java-based network-centric applications allow organizations to build and customize business applications that reach a wider audience much more quickly and with fewer support issues. In this case, Java truly gives a "write once, run anywhere" capability.

CBSS which was developed by using Java language as a cooperative C/S system where the client system can be accessed through the Web, inherits the features of these technologies. Only one version of CBSS distributed through the Web but it can run anywhere (platform independent) without being installed. Each time this version is modified, the client always get the latest version. Implementation, upgrading, and maintenance problems are reduced substantially.

Chapter 4. CBSS System Design and Implementation

The design of the CBSS was inspired by the Anson and Jelasi [1990] negotiation process stages and by the Gulliver [1979] eight-step negotiation process model. The philosophy behind the CBSS design is to develop computer software to support a collective bargaining process, either in combination with face-to-face meetings, or in a distributed-synchronous meeting. CBSS is not designed to provide suggestions or solutions to the users but to provide process support, especially through remote communication, documentation handling, and process structuring.

CBSS, as a computer-based system, provides at least three advantages over a traditional bargaining process (face-to-face negotiation): parallel communication, a structured pattern of bargaining, and automatic documentation. Parallel communication allows two or more participants to send messages at the same time. The participants do not need to take turns contributing ideas for discussion. In a face-to-face meeting participants must take turns speaking, otherwise other participants will confuse to whom they should listen. In using CBSS, two parties may transmit messages at the same time, the system will take care of message delivery, and there is no interference among these messages. Parallel communication promotes idea generation by allowing participants to contribute their ideas freely and anonymously. Two teams of negotiators using two computers located in two different places, are able to exchange messages without knowing who actually typed the messages and who

actually originated the messages. All they know is that the message is from the other side in the negotiation. Individual members of a team located at different sites may prepare messages privately among their own team members or send messages to a mediator or opposite team members. In a face-to-face meeting, sometimes a participant does not want to contribute an idea only because this particular person is afraid of being recognized as contributing an idea which may put him/her in a difficult position.

CBSS provides a structured pattern of bargaining, for example, by organizing the process from beginning to end according to the stages in the bargaining process, and by presenting each bargaining issue as a separate window among other organized windows. Each party can move, without losing the current status of each issue, from one issue to another. CBSS records all discussions and comments, so the participants can decouple themselves from a discussion to pause, think, type comments and then rejoin the discussion. In a face-to-face meeting, a participant must be present in the meeting and listen with full attention in order to capture the other participants' arguments. In using CBSS, a team may conduct a caucus meeting without being watched by the other team, and without losing any messages sent by the other team. In a face-to-face meeting, it may cause confusion if a participant is talking about one particular issue and the other participants are delivering messages about another issue. In using CBSS this situation may occur without confusion, because CBSS will deliver any messages to the place they belong. Structuring the process also makes negotiations easier to organize. For example, discussions on each issue are placed in a separate file and window, so that the negotiators can review its contents easily. Using a tape recorder to record all discussions is an example of unstructured process support, where finding a specific sentence belonging to an issue discussed is not an easy job.

Documentation support may also reduce communication problems that occur when participants miss the other party's points or they forget what has been discussed previously. This automatic documentation makes the contract agreement easier to create at the end of the negotiation process. In face-to-face negotiations, a speaker may be distracted if a participant suddenly asks the speaker to repeat a statement which was delivered hours ago. With CBSS, a participant may review all statements delivered before without disturbing any of the other participants.

The traditional bargaining process requires participants to be present in the same place at the same time for face-to-face negotiations. CBSS provides the possibility of conducting distributed-synchronous bargaining, or to support the traditional bargaining process. The negotiating parties may stay at their own place of work to conduct bargaining, thus reducing the cost of hotel rooms, conference rooms and travelling.

The system design and implementation of CBSS has followed a life-cycle of :

(1) System Specification : specified the most important requirements of the CBSS.

(2) General Design : specified the main components of the CBSS structure.

(3) Implementation : included repeated cycles of Programming, Debugging and Testing, and Modification steps.

4.1 Usability Issues

The revolution in personal computers and decreasing hardware costs make personal

computers accessible to a broader group of users, from novice users to computer experts, and from children to adults. The availability of software to support almost every human activities makes computers become useful and interesting, creating a tremendous demand for them. However, without attributes such as: ease of use, interesting interfaces, effectiveness, affordability, etc., the acceptance of computer systems by non-expert users would be more difficult even impossible.

Jacob Nielsen [1993] uses the "usability" term to represent some attributes associated with the software user interface as a part of system acceptability. According to Nielsen, there are five attributes that define software usability:

> "(1) *Learnability*: The system should be easy to learn so that the user can rapidly start getting some work done with the system.
>
> (2) *Efficiency*: The system should be efficient to use, so that once the user has learned the system, a high level of productivity is possible.
>
> (3) *Memorability*: The system should be easy to remember, so that the casual user is able to return to the system after some period of not having used it, without having to learn everything all over again.
>
> (4) *Errors*: The system should have a low error rate, so that users make few errors during the use of the system, and so that if they do make errors they can easily recover from them. Further, catastrophic errors must not occur.
>
> (5) *Satisfaction*: The system should be pleasant to use, so that users are subjectively satisfied when using it; they like it."

During the design and implementation process of CBSS, the usability issues were considered as the highest priority because no matter how sophisticated the system is, it has no value if users do not want to use the system. How well the CBSS prototype meets Nielsen' s usability specifications will be addressed in Chapter 5.

4.2 System Specifications.

As a negotiation support system which can be accessed on the Web, CBSS is required to be easy to use software, architecturally neutral, and must support the negotiation process from start to end. CBSS is different from existing NSS in several characteristics:

(1) CBSS is a process support NSS, and not a solution driven NSS.

(2) CBSS is designed to support dyad group meetings, so that it is different from the existing EMS types which support uniform group discussion.

(3) CBSS utilizes widely available computer networks such as the Internet in a "teleconferencing" setting. In this way CBSS differs from the existing process-support NSS which requires a special setting such as a "decision room".

(4) CBSS is accessible through the Web by using a Web browser. In this way CBSS can be accessed by anybody anywhere around the world.

(5) CBSS was developed with the Java programming language, so the application inherits the free-platform characteristics of Java. The software can be installed on many hardware platforms without re-writing and re-compiling the program.

(6) CBSS requires no software installation on the user' s computer, because the software is downloaded automatically from a Web server.

The most important system requirements for CBSS are defined as follows:

- CBSS should facilitate direct communication among the negotiators across the Internet.
- CBSS should provide documentation support for all discussions as well as private notes.
- CBSS should structure the negotiation process to make this process more productive.
- CBSS should be accessible from anywhere by any suitable machine connected to the Web.

Those system requirements can be broken down into the following specifications, including usability requirements:

To facilitate the direct communication among the negotiators,

- the system must implement a Graphical User Interface (GUI) standard,
- the system must provide the "What I See Is What You See" (WISIWYS) interfaces on public windows,
- the system must be able to provide parallel two-way communication between the parties,
- the system must provide coordination in ideas exchanges and message exchanges.

To provide documentation support,

- the system must have reliable data and file management support,
- the system must prevent most errors from occurring, and must provide easy recovery if errors do occur.

To structure the negotiation process,

- the system must provide user menu,

- the system must be specially designed for the negotiation process.

To be accessible from anywhere,

- the system must be able to utilize existing computer networks.
- the system must be executable under a Web browser,
- the system must be implemented as a Client/Server system.
- the system must be an architecturally neutral system.

4.3 The General Design.

Based on system specifications, the architecture of the CBSS was based on a cooperative processing Client/Server architecture, where the processes are divided between the client program and the server program. The CBSS server program is installed on a Web server site, and the client program, which is executable under a Web browser, is delivered to the user' s machine whenever the user accesses the CBSS web site. Server and Client programs were written in the Java language to ensure the platform-free properties. The "Abstract Window Toolkit" (AWT) package of Java provides the Graphical User Interface (GUI) and the What I See Is What You See (WISIWYS) property of the system' s windows.

The client program consists of three main components: **Dialogue Management**, **Data Management**, and **Communication Management**, to support two main modules: **Pre-Session** and **Session**. The Server program consists of three main components: **Client Management, File Management,** and **Communication Management**. The overall

architecture is shown in Figure 4.1A and Figure 4.1B. Figure 4.2 shows the CBSS home page, and Figure 4.3 to 4.5 show the interfaces when users start the program. Note: the figures for this chapter appear on pages 72 - 91.

Two modules providing direct interaction between the users and the system are Pre-Session and Session modules. These modules are presented to users as pull-down menus which are easy to use. The **Pre-Session** module (its architecture is shown in Figure 4.1B and its interface in Figure 4.6) is provided in order to support the preparation stage as well as the bargaining session stage. It supports recording the bargaining items (Figure 4.8 and Figure 4.9), setting the initial proposal for these items (Figure 4.10), and preparation of discussion notes (Fig 4.11). Related data are sent to the server to be saved as data files. These data files are used later during bargaining session stages.

Structuring of the negotiation process is implemented in the **Session** module where the process is divided into three phases (Figure 4.7): General Discussion, Issue Discussion, and Completing the Agreement. In each phase, a new window is created, which is divided into three parts: **Our Window**, **Their Window**, and **Common Window**. **Our Window** displays messages which are sent by the user to other parties. **Their Window** displays messages which are received from the other side. The **Common Window** is a window for displaying the agreement among the negotiating parties. A small dialogue window, activated by a **Compose** button (Fig 4.14), is called the Comment Editor (Figure 4.15). This is provided to prepare messages, either for public or for private use (e.g. to save the message as a note). In discussing a topic, users contribute ideas which are sent to public windows (other side' s

Their Window and the user' s **Our Window**) and at the same time recorded in text files at the server site. These text files are the discussion log-books, and can be retrieved to re-start negotiations in case of scheduled breaks or system outages.

The structure of the negotiation process is in parallel with Gulliver' s [1979] negotiation process stages. **General Discussion** supports Agenda Setting (step 2) where parties discuss the collective bargaining agenda until they agree on the issues to be negotiated, the time allocated to each issue, and the trade-off of issues. Step 3 (Explore the Field) is also supported by the General Discussion phase where parties collaborate to limits the issues in dispute. **Issue Discussion** supports step 4 (Narrowing the Difference), step 5 (Preliminaries to Final bargaining), and partially step 6 (Final Bargaining), where parties negotiate and search for solutions to a particular issue. **Completing the Agreement** supports step 6 (Final Bargaining) and step 7 (Ritual Affirmation), by providing a communication channel for proposal exchange and the documentation of the final agreement.

Even though CBSS' s process structuring does not precisely follow Anson and Jelassi [1990] process stages, the pre-session module was inspired by their work. In addition, there are function similarities between CBSS' s session module and Anson & Jelassi' s session stage. For example, CBSS' s General Discussion module is similar to Setting the stage and Formulating the problem, Issue Discussion is in parallel with Processing the issue stage, and Completing the agreement is in parallel with Resolving the issue stage.

Three system components are designed to support user activities during the Pre-

Session and the Session: Dialogue Management, Data Management, and Communication Management. The **Dialogue Manager** manages the interaction between users and the system. This subsystem provides interfaces for the users, such as: User Menus, Comment Editor, Hot Line, Form-fill, and various windows to accommodate message exchange during the negotiation. These interfaces are activated with keyboard-input or mouse clicks. User Menus are the menus provided to the user, which are: Pre-Session Menu, Session Menu, and Help Menu. Each menu is a pull-down menu, consisting of several sub-menus. Comment Editor is a small window provided to the users where they can edit their messages or comments before sending these messages to the other users. The Hot Line (Figure 4.4) is a dialogue tool for the users, especially for short, important, and immediate messages. It is also a simple monitoring window to display activities of the other side so actions can be synchronized during the Session. Form-fill is a kind of dialogue form provided to the users in order to enter their group-name and password, to select their team' s name, and to enter bargaining items manually.

The **Data Manager** manages the data created during the usage of the program. Each dialogue window is associated with one text buffer. Each time the user types a message or a note, the Data Manager will put the data into the right buffer, and each time a window is refreshed, it will select the right buffer to be displayed on that window. The Data Manager also intercepts incoming messages from the Communication Manager to be recorded in the right buffers.

The **Communication Manager** manages data communications through the Internet. The main functions of this component are: to receive messages from the server and to deliver

these messages either to the Dialogue Manager or Data Manager, and to send messages received from either Dialogue Manager or Data Manager and to deliver these messages to the server.

The Server program consists of three main components: Client Management, File Management, and Communication Management. The **Client Manager** manages the client database, the **Client Identification** (submodule of the Client Manager) identifies the users as management, union, or mediator, checks the incoming passwords, and provides correct data paths to each client. The **File Manager** manages the creation, retrieval, and updating of data files. The **Communication Manager** manages data communication between the client and the server and among the clients. It receives messages from the client, provides these messages to File Management, and sends back messages to other clients.

Several modules in the Client system are designed specifically for CBSS such as: Note function, Hotline, and Hotline Watcher. The Note function provides a convenient way to keep users' notes as private notes. Negotiators are allowed to prepare ahead as many notes as they wish, to be accessed online later during the negotiation process. The contents of these notes can be varied from previous collective bargaining agreement, industry objective data, to bargaining strategies. During the negotiation these note can be viewed, copied, or sent as part of discussions. Hotline is useful for exchanging short messages or for notifying other negotiators about actions that will be taken by a user. Hotline watcher is a blinking note to inform users that a new message is coming into the Hotline window, so that the users may turn their attention to the Hotline.

4.4 Implementation.

Microsoft Windows 95(tm) was selected as the environment for the CBSS development since Windows has become a technical standard in GUI design and it provides a multi-tasking environment. The World Wide Web was selected as the networking platform because of its popularity and its reach. The system program is written in the Java programming language (Sun MicroSystem' s JAVA SDK version 1.0). The JAVA language was selected because: 1) HTML handles only static displays, but Java is needed to accomplish dynamic tasks, and 2) Java programs are architecturally neutral. The advantages of using the Web as a networking platform and Java as the development programming language were mentioned in Chapter 3. The implementation of the user interface is shown in Figures 4.2 to 4.19 (page 75 to 92).

CBSS implementation concentrates on facilitating the negotiation process, but it does not provide any suggestions or solution alternatives to the users. Since CBSS runs under Windows 95, Windows NT, or other multi-tasking environments, any DSS packages which run in the same environment may be used to provide suggestions and solution alternatives to the users in conjunction with CBSS.

CBSS programming is presented in Section 4.5 which outlines CBSS programming principles. More detailed descriptions appear in the appendices. Debugging and program testing were done each time the program was modified. Two Ph.D students from the Human Resources Management and Labour Relations Department assisted as test users during these

periods. Suggestions, critiques, and bugs were recorded during these tests and were used to modify the program. Programming, debugging, testing, and modification steps were repeated several times during the implementation stage until the system satisfied system specifications. CBSS also evolved gradually during development from a direct computer connection, to a local area network environment, then to an Internet (non-Web) platform, and finally to to a Web platform.

The Server system is implemented as a stand-alone Java program and it is installed on a Web server machine. The Client system is implemented as a Java applet which is put under the CBSS Web' s home-page. The Client system will be downloaded automatically whenever a user accesses this home-page. The mediator function is already implemented in the Client system, so that the same Client system will be downloaded by a union member, a management members, or a mediator. Unfortunately, the mediator function was not tested during the experiment.

4.5 Programming.

A program written in the Java language is either a Java applet or a Java application. Java applets are programs that can be downloaded over the World Wide Web and executed by a Java-capable Web browser on the user' s machine. Java applications are more general programs like other programs written in more conventional programming languages. They do not require a Web browser to run, but they can not be downloaded over the Web. Java

applets have some limitations which should be considered when they are designed. Some of these limitations are as follows:

(1) Some browsers do not allow applets to read from or write to the client' s file system.

(2) Applets may communicate only with the server where the applets are originated.

(3) Applets can not run any programs on the client' s file system.

(4) Applets can not load programs native to the local client platform, including shared libraries such as dynamic link libraries (DLLs).

These Java Applet limitations were the main challenges in developing the CBSS program. CBSS must maintain documentation of all data generated during a bargaining session, but the browser does not allow the program to access local files. As a consequence, CBSS is implemented in both program categories. The server system is a Java application while the client system is a Java applet. The client system gathers the data and sends it to the server system. The server system captures the data sent by the client, and saves it in the appropriate files.

The difference between defining a Java application and defining a Java applet is shown as follows:

```
// the Server program definition--an application
class Server {
      public static void main (String args[]) {
      .....
      }
}
```

```
// the Client program definition--an applet
public class Client extends java.applet.Applet {
        public void init() {
        .....
        }
}
```

Since Java is a true object-oriented programming language, a program file must contain one or more class definitions. A class contains data definitions and methods that manipulate the data. An object is an instance of a class. Java provides a comprehensive library which contains many classes that are ready to use in developing a program. These classes are organized into packages in order to be easy to use. Some of the most important packages used in developing the CBSS program are listed as follows:

(1) java.lang.* : contains classes and interfaces that make up the core of the Java language.
(2) java.util.* : contains various utility classes (i.e., Date, Hashtable, Vector) and interfaces (i.e., Enumeration, Observer).
(3) java.io.* : provides input and output classes and interfaces for streams and files.
(4) java.net.* : provides classes and interfaces for performing network operations.
(5) java.awt.* : the Abstract Window Toolkit, contains classes and interfaces for windows operations.
(6) java.applet.* : contains classes and interfaces for applet specific behavior.

These packages must be imported into the program before using them. For example the client system program is started as follows:

```
import java.awt.*; //  we need to create and to manipulate windows
import java.io.*;   //  we need certain input and output operations
import java.net.*;  // we need network operations
import java.util.*;  // we use some utilities
```

Some basic programming techniques used repeatedly in the CBSS system are

presented in Appendix C. The program flowcharts are presented in Appendix A. The Client System and the Server System programs skeleton are described in the following paragraphs.

The Client System consists of three classes which are:

- **CBSS class** which is the main program of the Client system,
- **Frame class**, objects constructor class includes: Dialogue manager, and Data manager,
- **StreamListener class**, class that handles communication (Communication manager).

These classes are supported by 28 functions as follows:

(1) **handleEvent()** - user interface main function, handles all mouse clicks.

(2) **keyUp()** - user interface, handles keyboard keys.

(3) **SetItems()** - reads bargain data in or view/edit bargain data.

(4) **ListItems()** - lists bargain data.

(5) **DeleteItem()** - deletes one bargain item.

(6) **InsertItem()** - inserts one bargain item.

(7) **createIssList()** - creates the list of bargain items.

(8) **GetBargainTxt()** - asks the Server to send bargain data.

(9) **parseList()** - receive the bargain data from the Server and convert them into a list.

(10) **BargainUpdate()** - update the bargain items list.

(11) **PrepareNote()** - creates window for preparing a note.

(12) **GetTheNote()** - asks the Server to send notes name list.

(13) **NoteName()** - create local notes name list.

(14) **GetNoteContent()** - asks the Server to send a note' s content.

(15) **NoteContent()** - matches note name with note content.

(16) **DisplayNote()** - displays note contents (view/edit a note).

(17) **GetPassword()** - asks the Server to check user' s password.

(18) **ReadGeneral()** - creates and displays General Discussion windows.

(19) **SelectOneIssue()** - displays issue list and waits for user' s choice.

(20) **DiscussIssue()** - creates and display Issue Discussion windows.

(21) **Agreement()** - creates and displays Agreement windows.

(22) **ComposeText()** - creates and manages the comment editor.

(23) **SendToPublic()** - send the messages/comments to the Server.

(24) **OpenCommon()** - handles the common windows.

(25) **Post()** - sends common window' s contents to the Server

(26) **RefreshWindow()** - refresh discussion windows whenever a message is received.

(27) **SaveThisFile()** - save the contents of common window into a file.

(28) **HelpDisplay()** - displays help files.

The Server System consists of three classes:

- **Server Class** - the main program and includes File manager and Client manager.
- **Connection Class** - handles the communication (communication manager)
- **Socket Class** - holds connections and includes communication support functions.

4.6 How to Use CBSS.

In the following paragraphs, the main steps to follow in using CBSS are described:

1. **Start the System:** Put your diskette into the drive and then point the mouse to the

Netscape Browser icon and double click the mouse' s left button. After activating this software, you may start a http connection to the CBSS site, which is "**http://pc-suarga-2.business.mcmaster.ca/cbss**", or you may select this site in the "bookmark" which is named "CBSS". The CBSS main page will be displayed, on the left hand side of the page there are four buttons (Figure 4.2):

(1) **Introduction**: a brief introduction to CBSS
(2) **Requirements**: explains the software and hardware requirements to run CBSS
(3) **Limitations**: some of CBSS limitations are presented here.
(4) **Start** : to start the CBSS program.

To start the program, click the "START" button, and within a minute the main menu bar is displayed on the top-left corner. A small window is displayed and asks you to enter your Group Name and your Password (Figure 4.3).

(1) Point the mouse to the space provided for the Group Name
(2) Type your group name, for example Group-1, and press Enter
(3) The HOTLINE (Figure 4.4) window will be displayed on the bottom right side.
(4) Type your password, (any password that is easy to remember), press Enter
(5) If this is your first time using CBSS, another small window will be displayed and asks you to select your team (Figure 4.5), either Management team or Union team.
(6) Click the mouse' s left button on the spot provided.

2. **Activate HOT-LINE:** the HOT-LINE window (Figure 4.4) is activated automatically as soon as you have entered your group name. This HOT-LINE window is displayed on the bottom right of the screen. You can use this HOT-LINE to send short messages to the other side.

3. **Prepare Bargain Items Data:** The first thing that you must do before starting the negotiation is to read the bargaining items data into the system. Put your diskette (the one with "bargain.txt" file) into drive A:.

(1) Open a wordprocessor program (e.g WP) and read your bargain data from your diskette.

(2) Highlight all of your data.

(3) Press the Right button of the mouse, and select **copy,** or click the **copy icon**.

(4) Close or minimize your wordprocessor window.

(5) Click **PRE-SESSION menu** (Figure 4.6), and select **Type Bargain Data in** from the menu, a small window is opened automatically (Figure 4.8), click this window with the mouse' s left button.

(6) Click the mouse' s right button, and select **Paste**

(7) Your data should be displayed now in Bargain data window

(8) Click the **Save** button.

If you haven' t prepared any bargain data on the diskette, you may start at step (5) and then "Type your data on that window". Click SAVE button before you leave the window. If both teams have not made agreement on what issues to discuss, go to step 7 using **General Discussion** window to set up the list of issues then go back to this step.

4. To make sure that the issue files are created properly, select **List Bargaining Items** from the **PRE-SESSION** menu and check the list (Figure 4.9). If this list is not correct according to your notes then ask for help.

5. You can select **View/Edit Bargain Items** (Figure 4.10) from the PRE-SESSION menu to check the issue attributes one by one, and make changes if necessary. Don' t forget to **close** this function whenever you finish.

6. Whenever your team is ready to negotiate, notify the other side by sending a message through the HOT LINE that your team is ready. Wait until the other side replies to your message (their reply will appear as the latest message as well as in the HOTLINE Log Book area of the window).

7. If your team and the other side are ready, then you can start the negotiation by clicking the **SESSION menu** (Figure 4.7). The **SESSION** full-down menu is displayed with three options: **General Discussion**, **Issue Discussion**, and **Completing the Agreement**.:

Select **General Discussion** function if your team has something to discuss about issues in general, such as the order of the issues to be discussed, time table and agenda, and so on. Please make sure that your team' s list of issues is in the same order with the other team' s list of issues, otherwise, you will be on different issues when you start discussing those issues later on. A General Issue Discussion window (Figure 4.14) with three parts is displayed on the screen. The top-left part is **Their Window** which displays the message sent by the other side to you. The top-right part is **Our Window** which displays the message you have sent to the other side. The bottom-left part is the **Common Window** which can be used to display agreements. For instance, in general discussion, you may write the negotiation agenda that both sides agree on. Click the **Compose Common Window** button so that a **Comment**

Editor is displayed (Figure 4.16). You can **copy** any text from Our Window and Their Window and **paste** it into this Comment Editor window. Do not forget to **save** it each time you finish making a modification. All three windows are used to keep track the communication between two parties. You cannot type a message directly in these windows, you must compose the message by clicking the **Compose** button.

You can start the discussion by sending a message. To prepare a message in a discussion session, use the mouse to point and click any area of Your Window, click the **Compose** button with the left button of the mouse. A small window will be displayed as a **comment editor** (Figure 4.15). You may read a note into this compose window or do direct typing in this window. To read a Note into the Compose window: a) click the **ReadNote** button, b) select the note' s name from the list (Figure 4.12), and c) click **OK** button. If you do **direct typing,** type the idea, comment or information in this small window, and press Enter key each time you reach the right margin, since there is no word wrap function. Click the **Send** button by using the mouse left button, to send the message to other side, or to the Common Window.

8. If both teams agree to discuss or make an agreement on the issue: Click **Issue Discussion** function from the SESSION menu. A small window for selecting the issue' s name will be displayed (Figure 4.16) on the top-left side of the screen. Select one issue from the list and then click the **OK** button. The Issue Discussion Window with three parts is displayed (Figure 4.18). It is organized in the same way as the General Discussion window.

Continue the issue discussion through **Our Window** and **Their Window**. The conclusion that both sides have reached can be put in the **Common Window**. To put conclusions or agreements in the Common Window, click **Compose Common Window** button, a Comment Editor is displayed (Figure 4.16). Type or Copy and Paste the conclusions on to this Comment Editor. Don' t forget to **Save** the common window contents each time you finish typing sentences into it. You can send the contents of the Common Window to the other side by clicking the **Post** button so that the other side can see it and make modifications if necessary.

9. If an agreement cannot be reached, you can leave the discussion for this issue and start another issue discussion. You can go back to this issue later on.

10. After solving all the issues, then activate the **Completing the Agreement** function. One team could take the responsibilities in preparing this agreement (Figure 4.19) while the other team should monitor and give comments.

11. At the end of negotiation, notify the other side through the **HOT-LINE**, and then click **Exit** from the main menu.

12. Maximize the Netscape Window and Select File/Close or Exit.

13. You can then print out the agreement and discussion logs from the files saved on the disk.

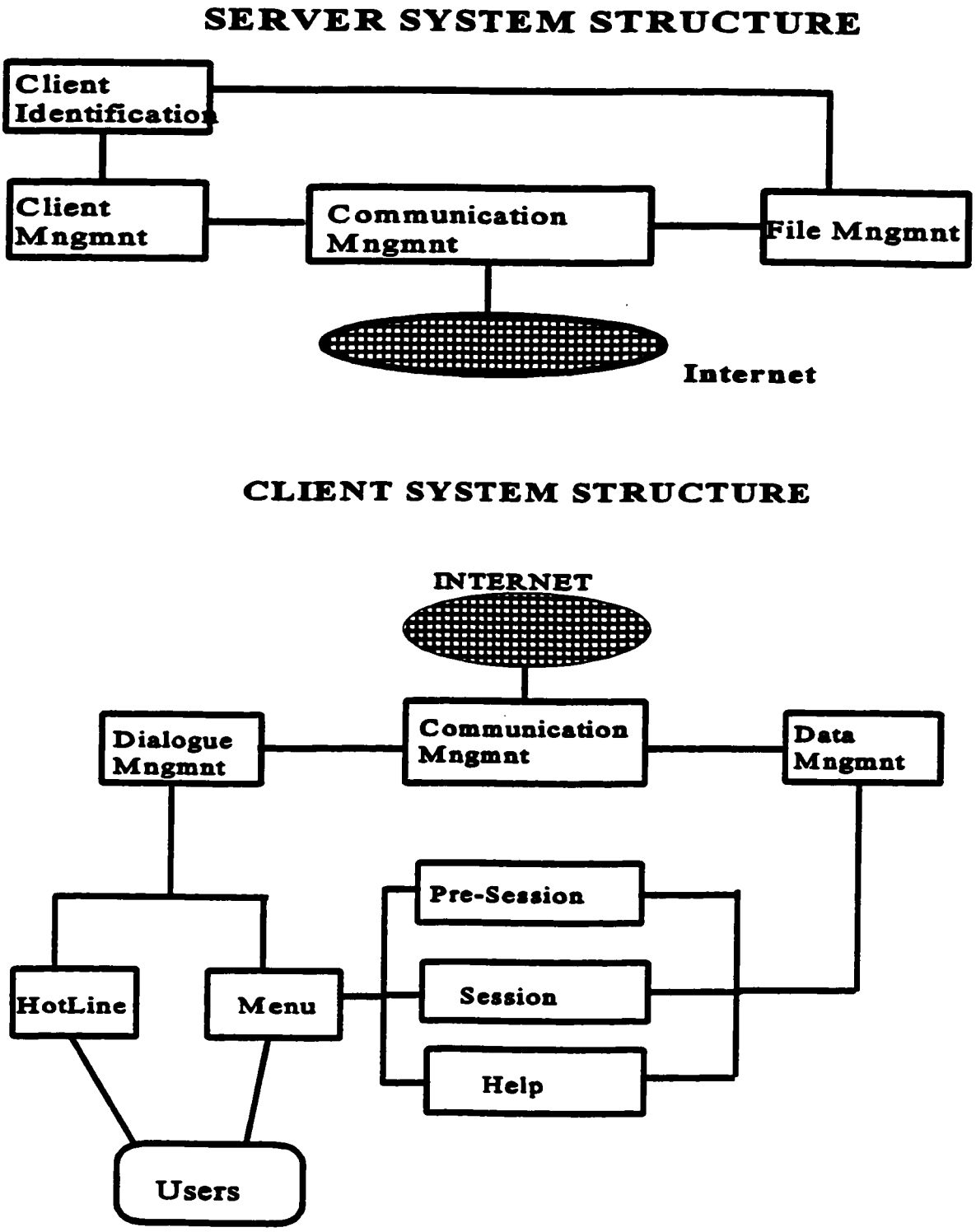
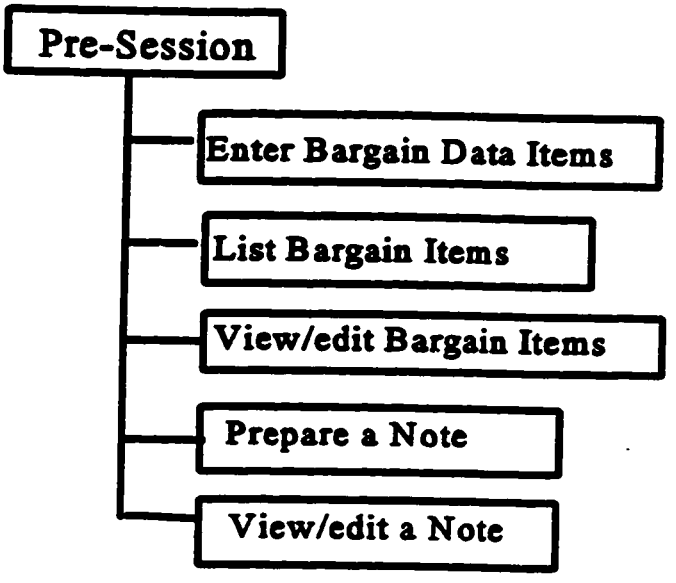


Figure 4.1A - CBSS SYSTEM ARCHITECTURE



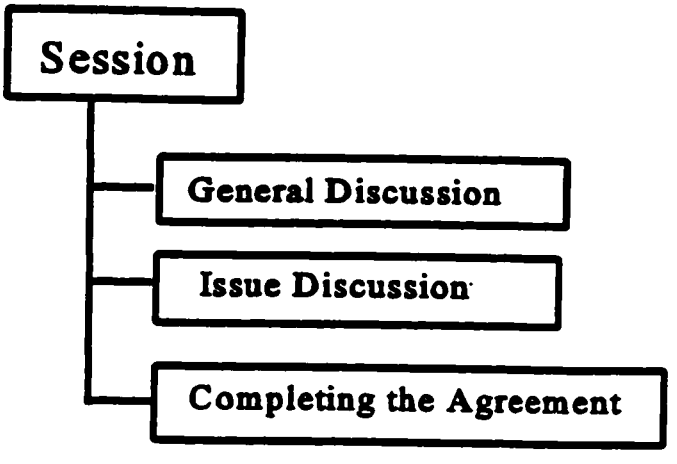


Figure 4.1B - Presession and Session Modules

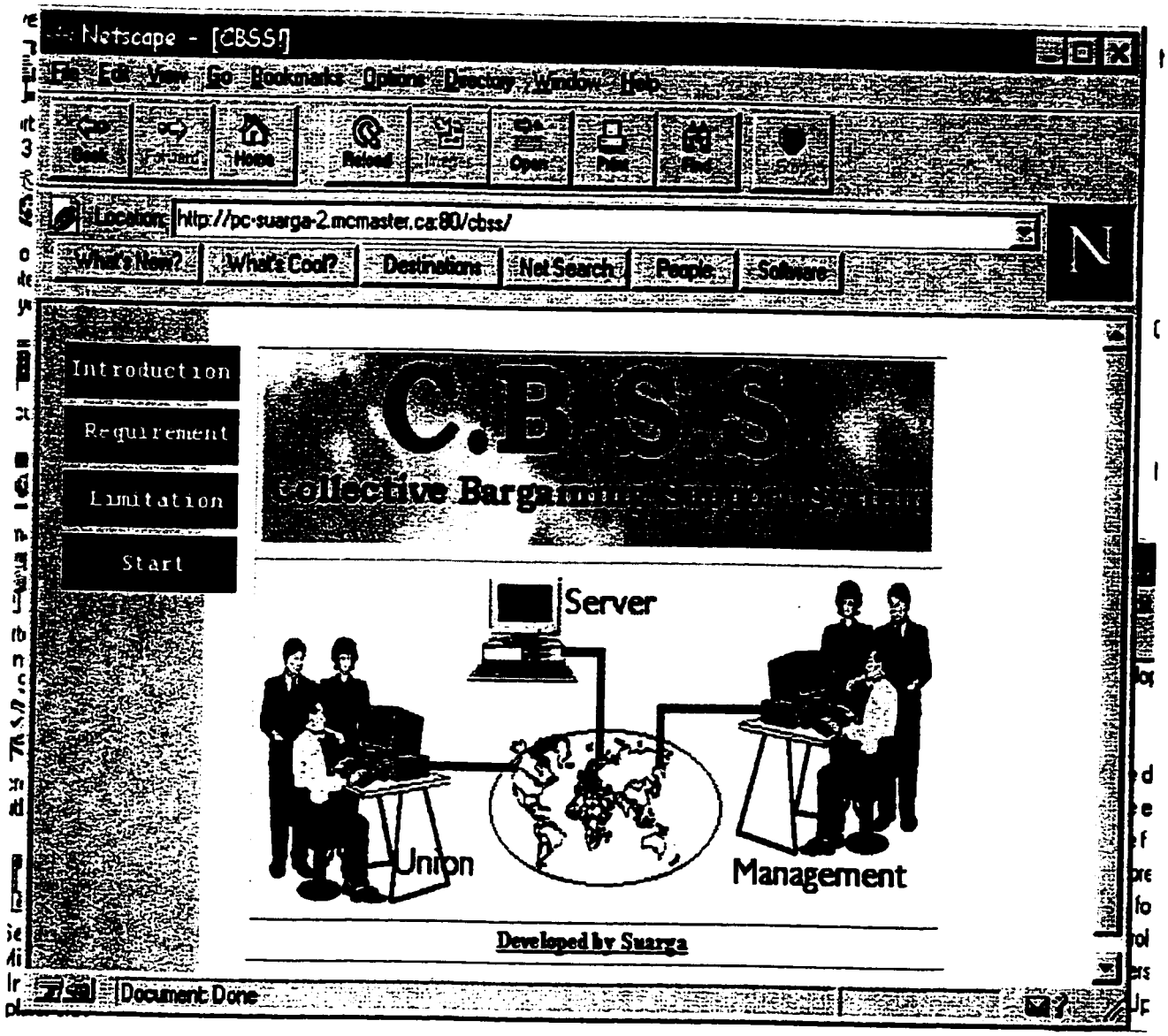


Figure 4.2 - CBSS Home Page.

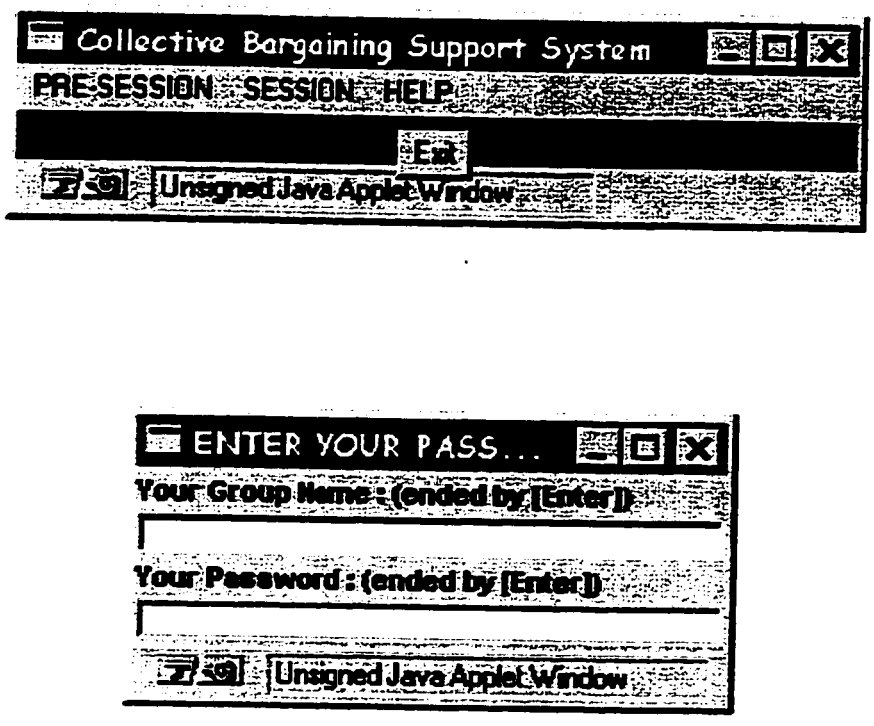


Figure 4.3 - Enter Group_Name and Password.

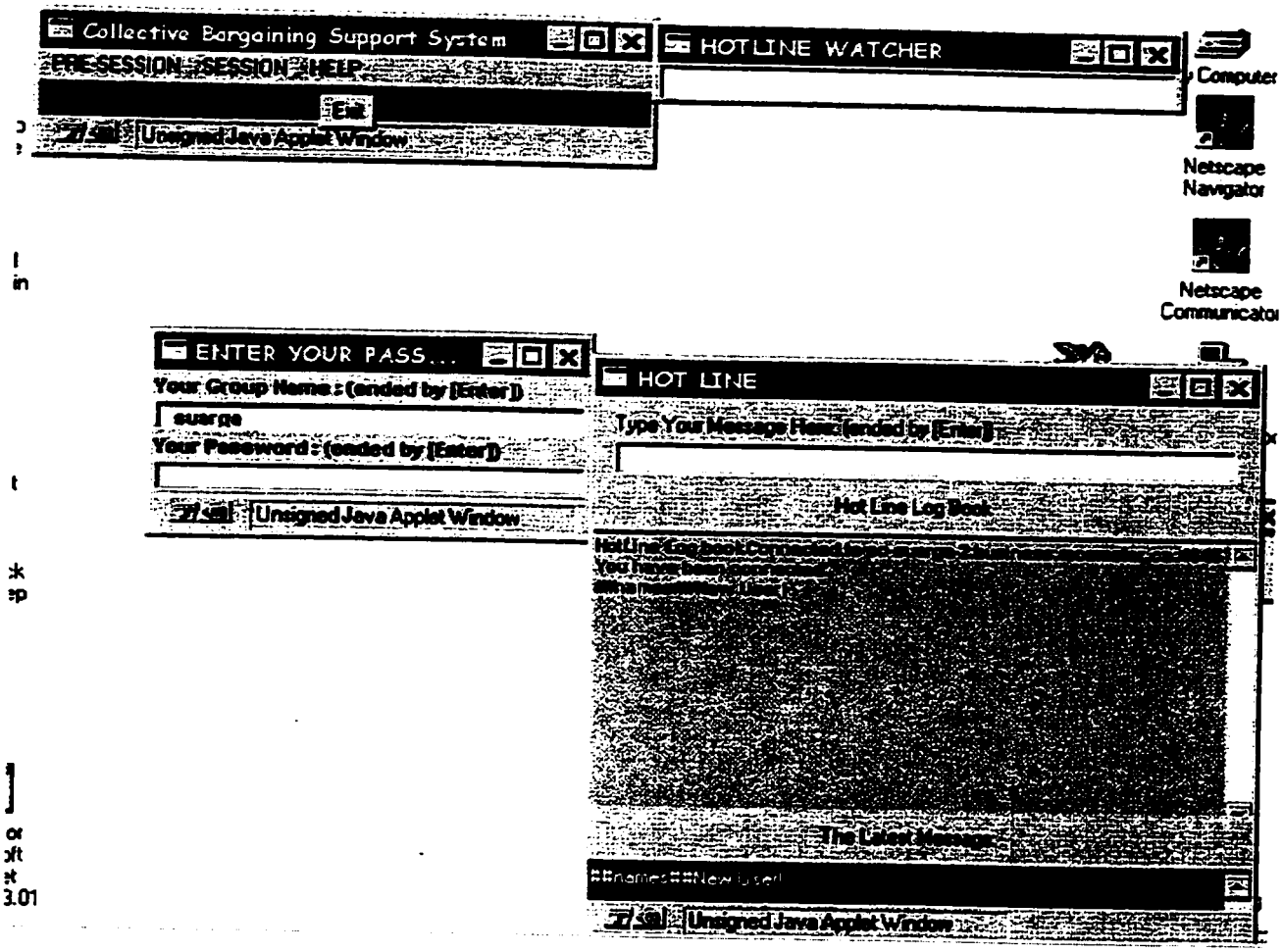


Figure 4.4 - HotLine and HotLine Watcher.

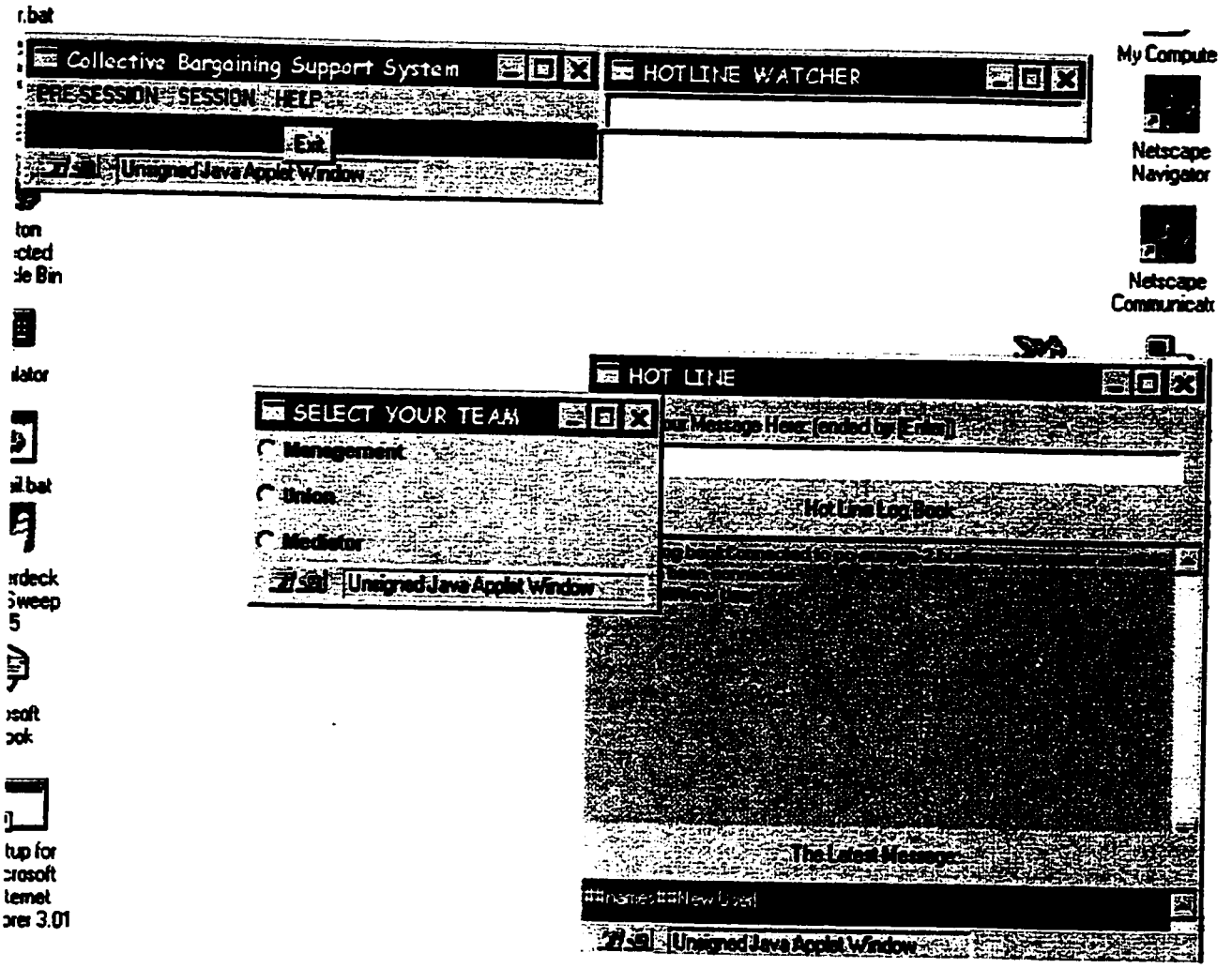


Figure 4.5 - Select Your Team (Management/Union/Mediator)

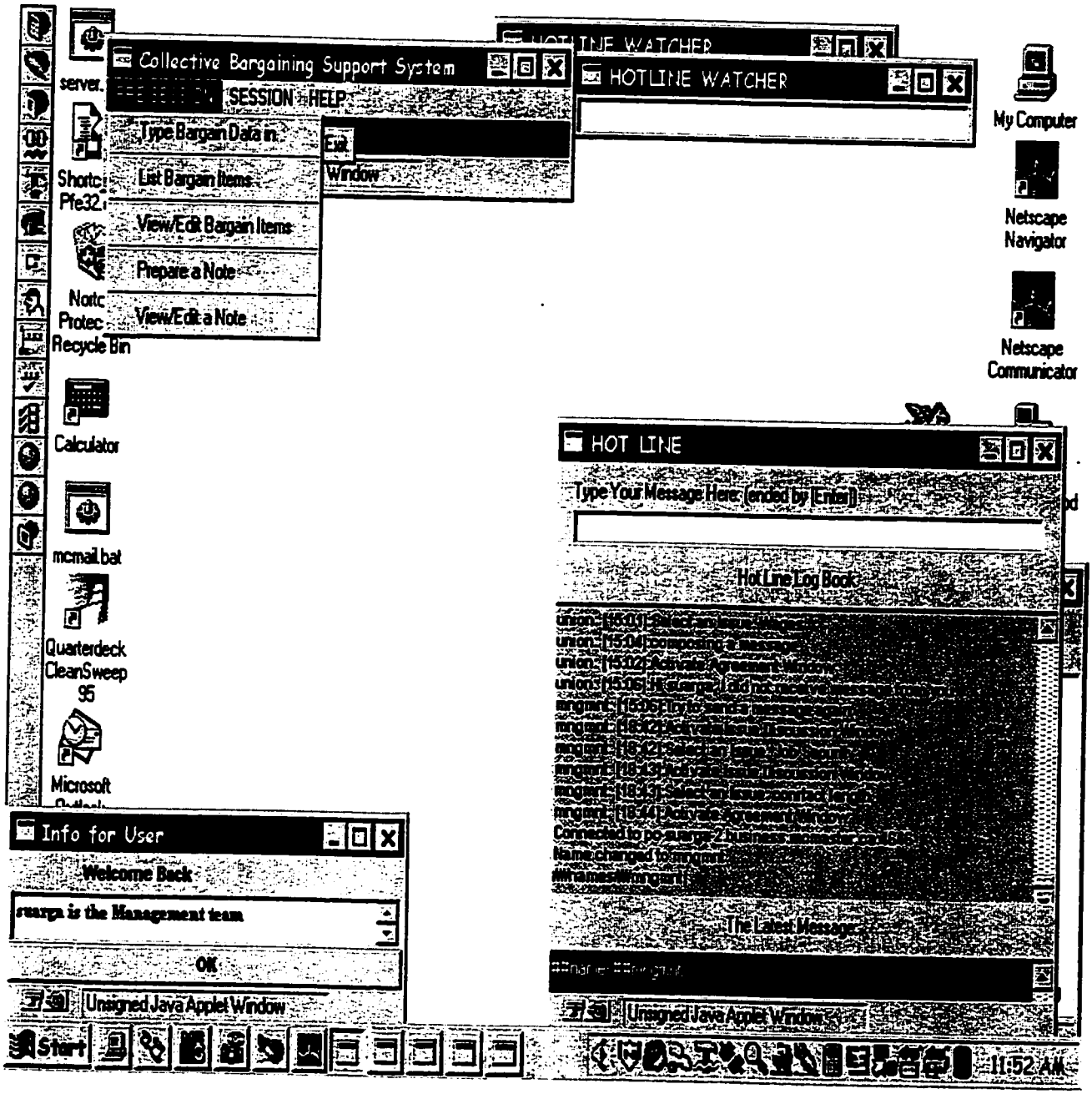


Figure 4.6- Pre-Session Menu & HotLine Window

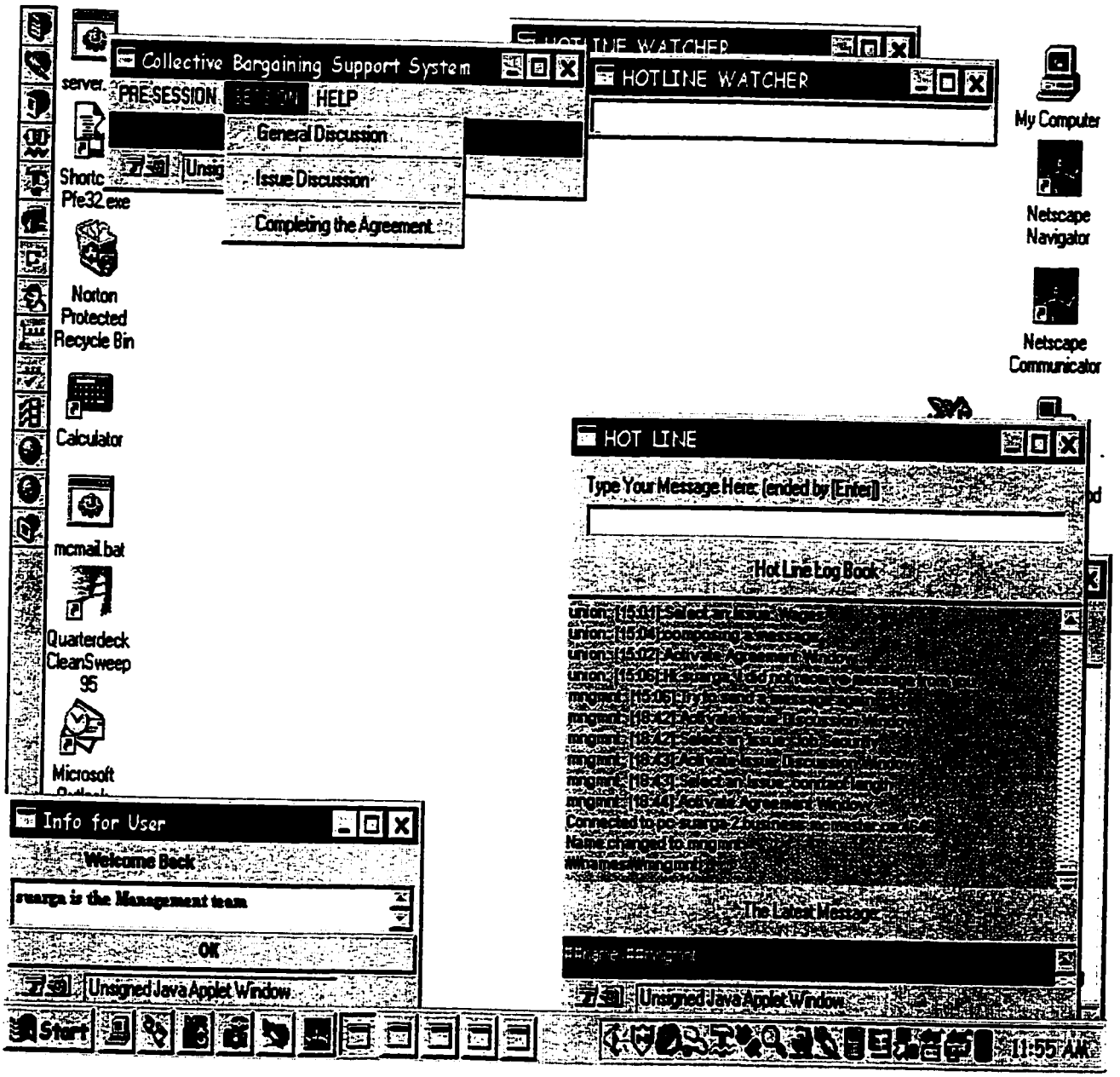


Figure 4.7 - Session Menu

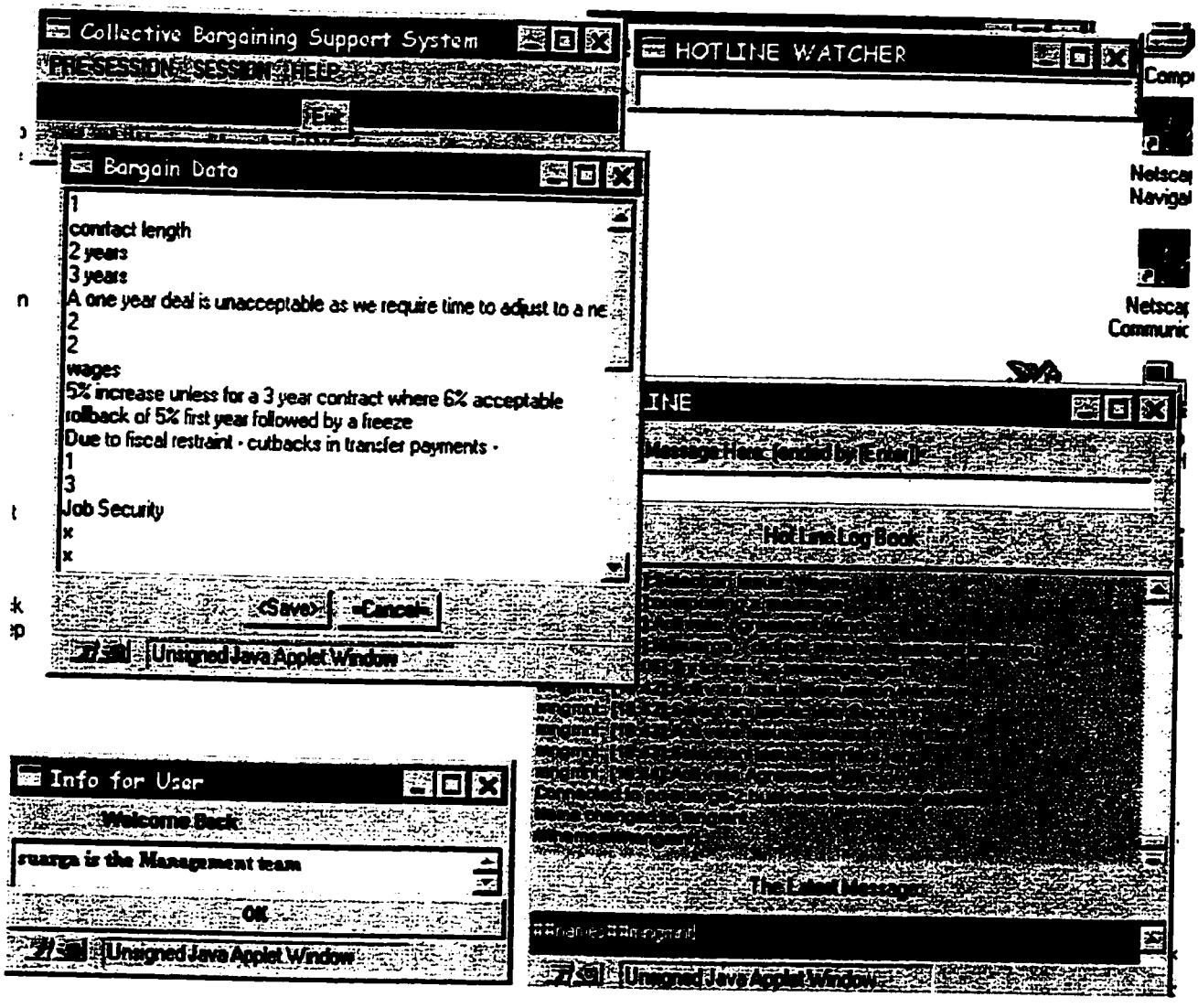


Figure 4.8 - Type Bargain Data In

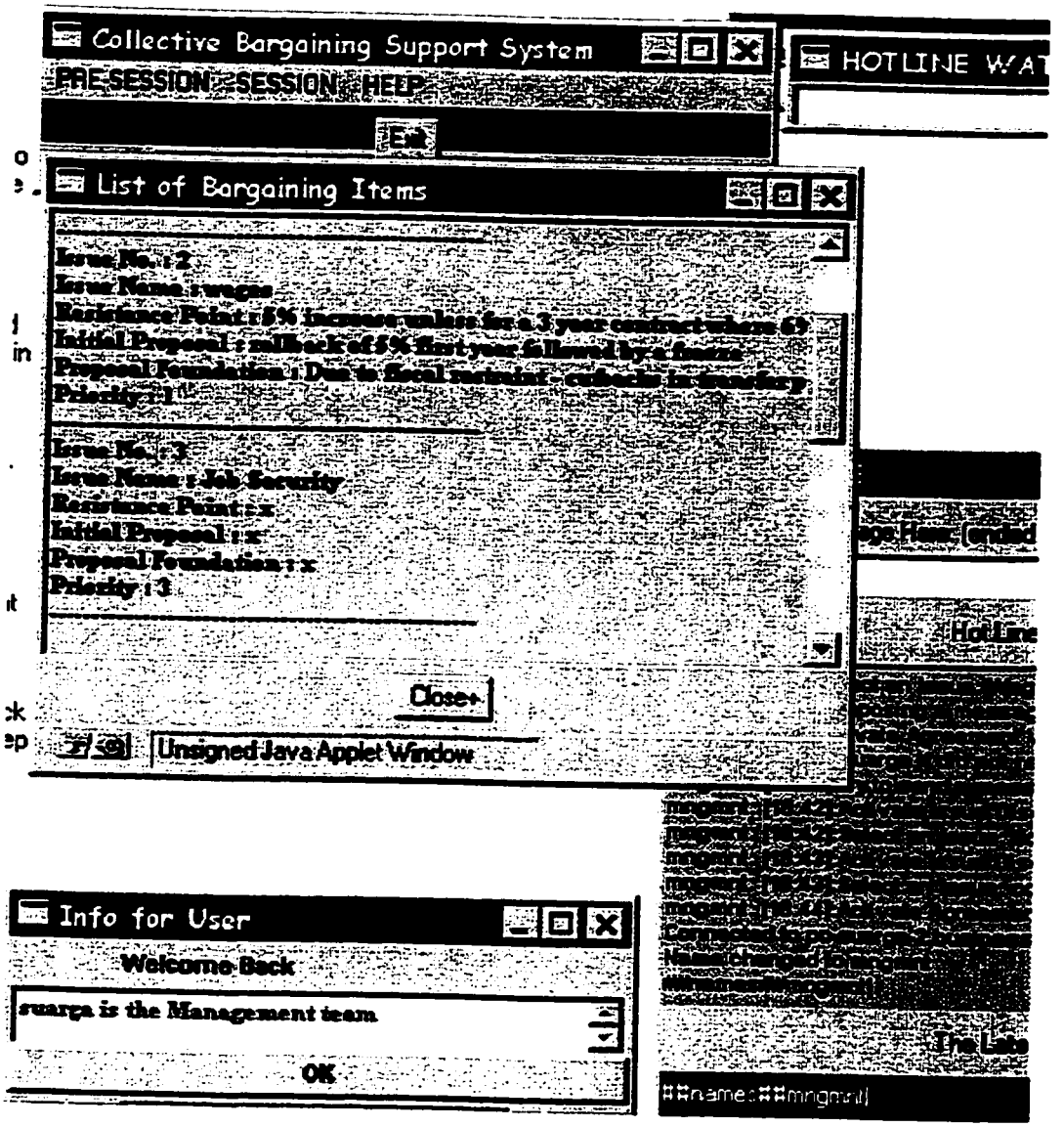


Figure 4.9 - List Bargain Items

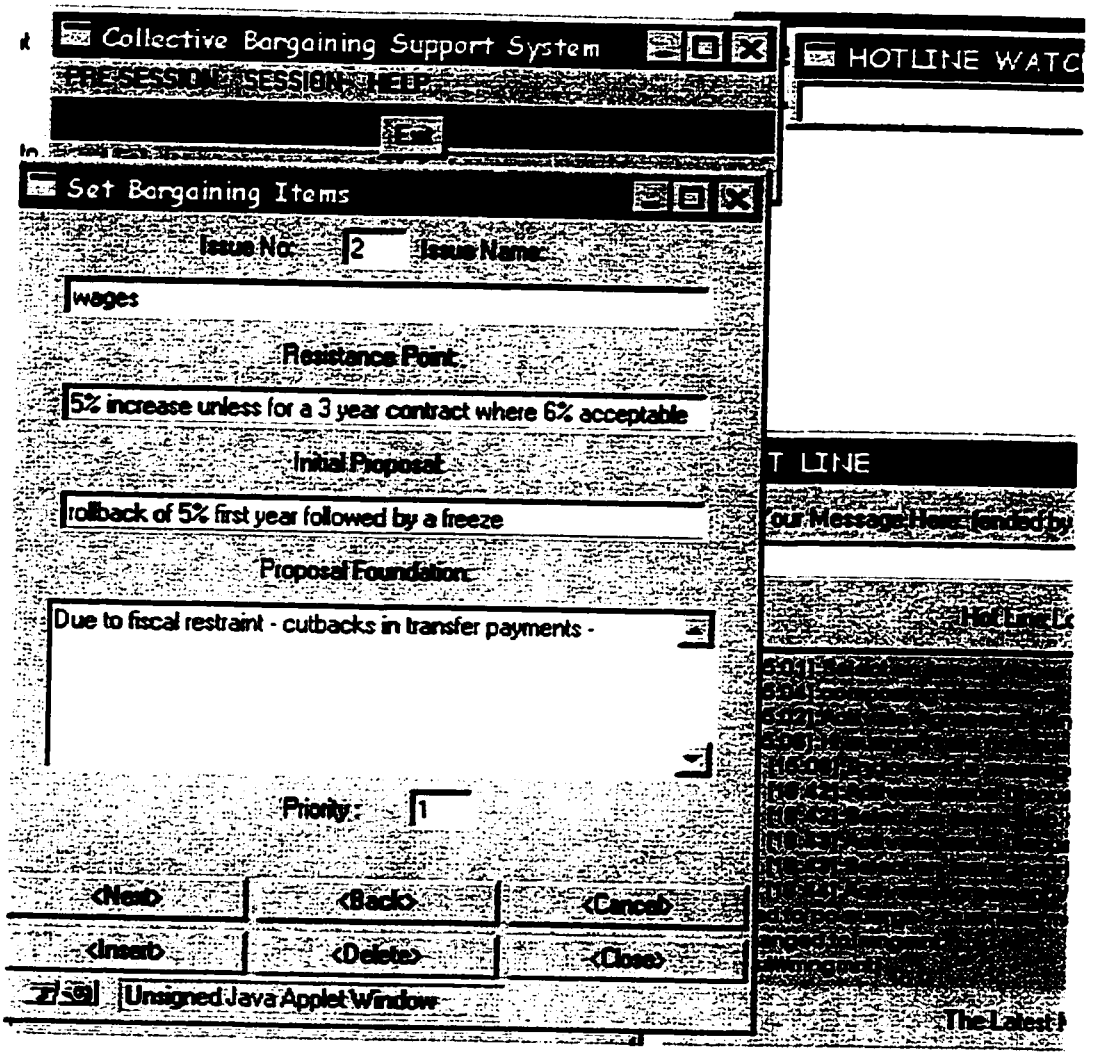


Figure 4.10 - View/Edit Bargaining Item

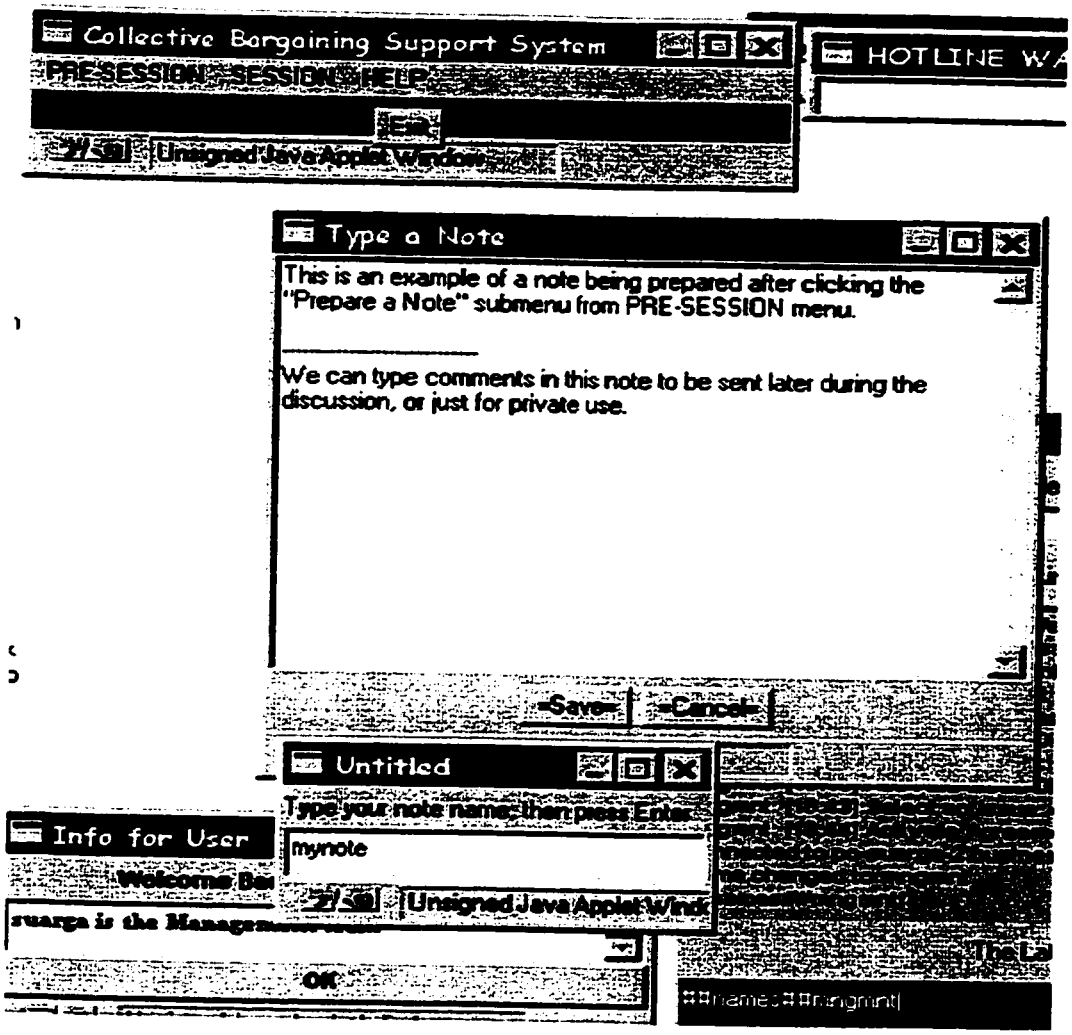


Figure 4.11 - Prepare a Note

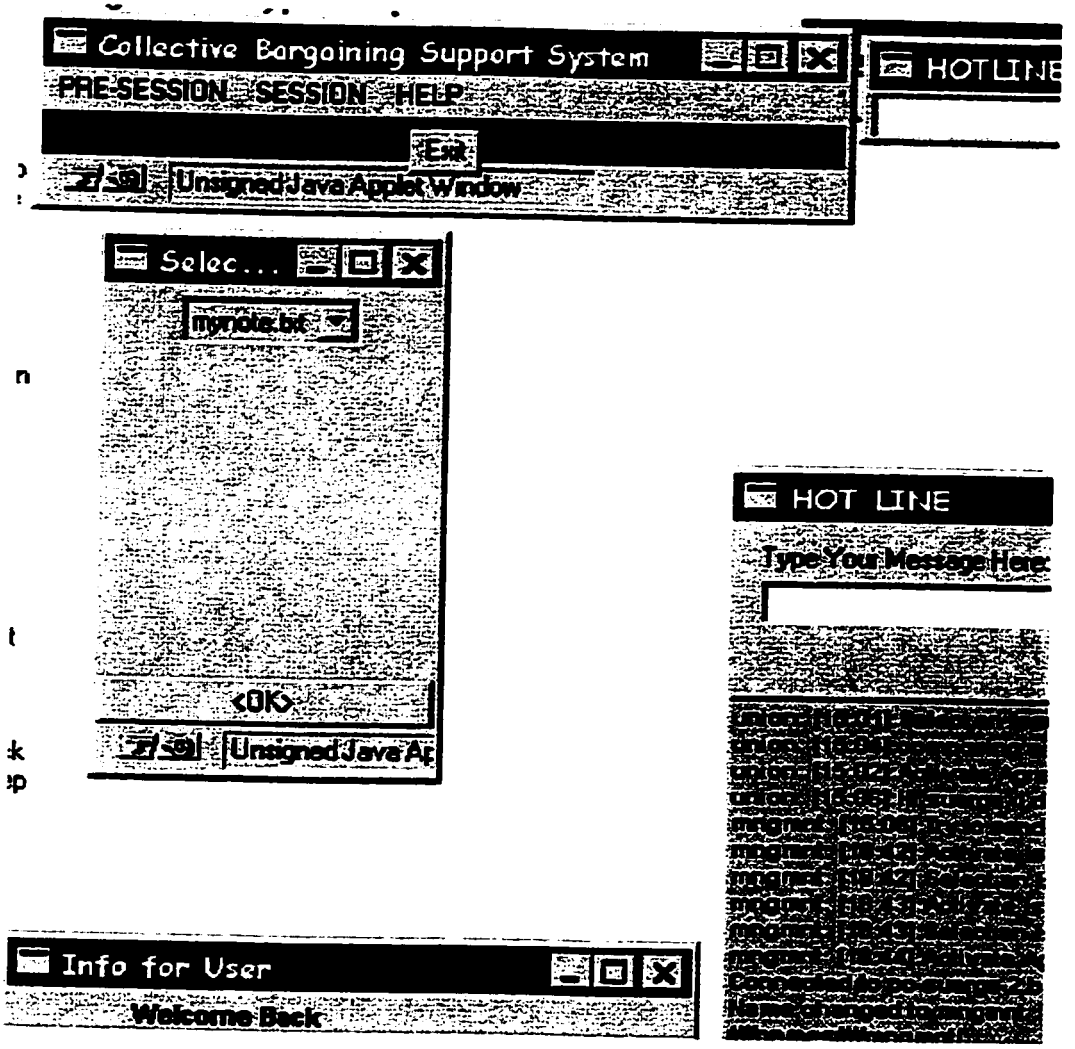


Figure 4.12 - Select a Note Name to be displayed

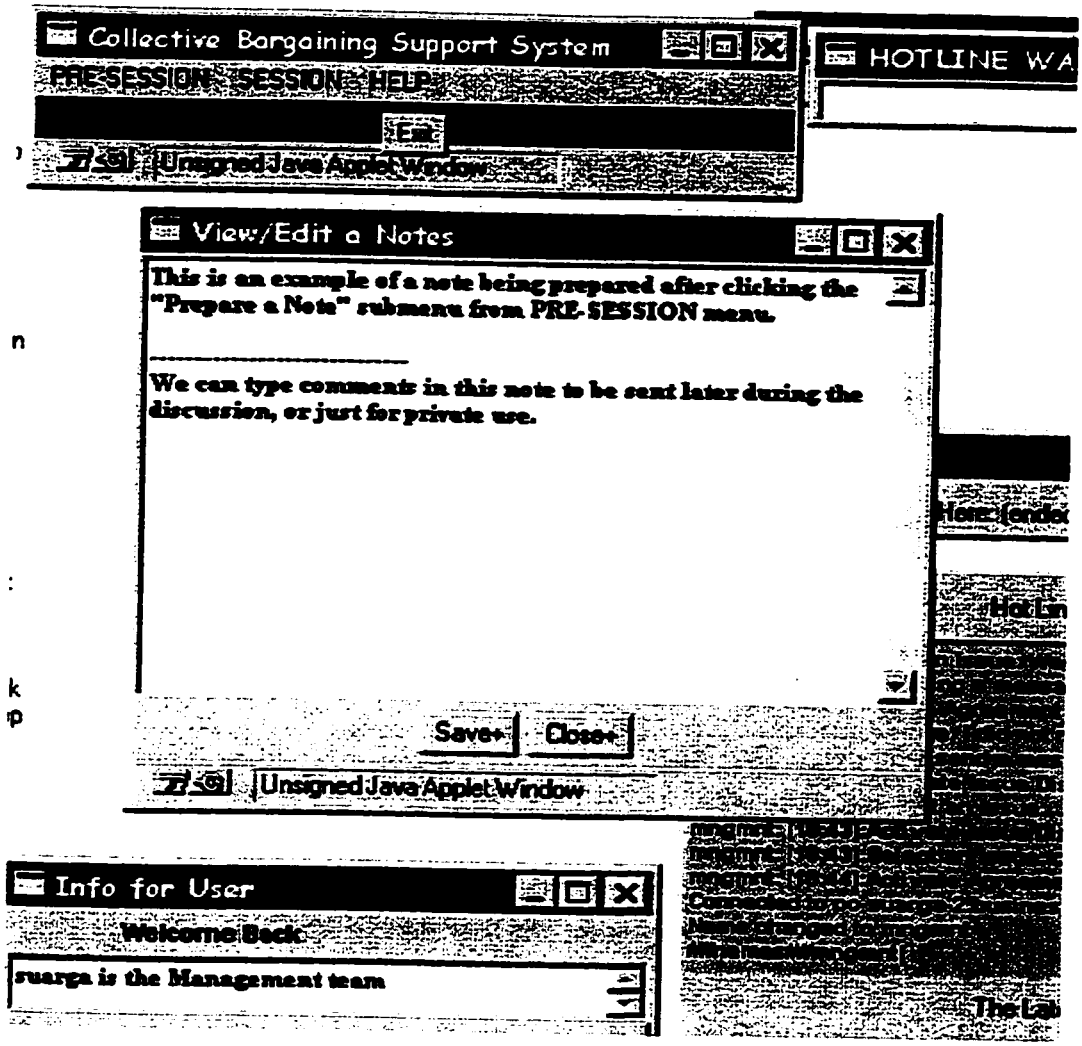


Fig 4.13 - View / Edit a Note

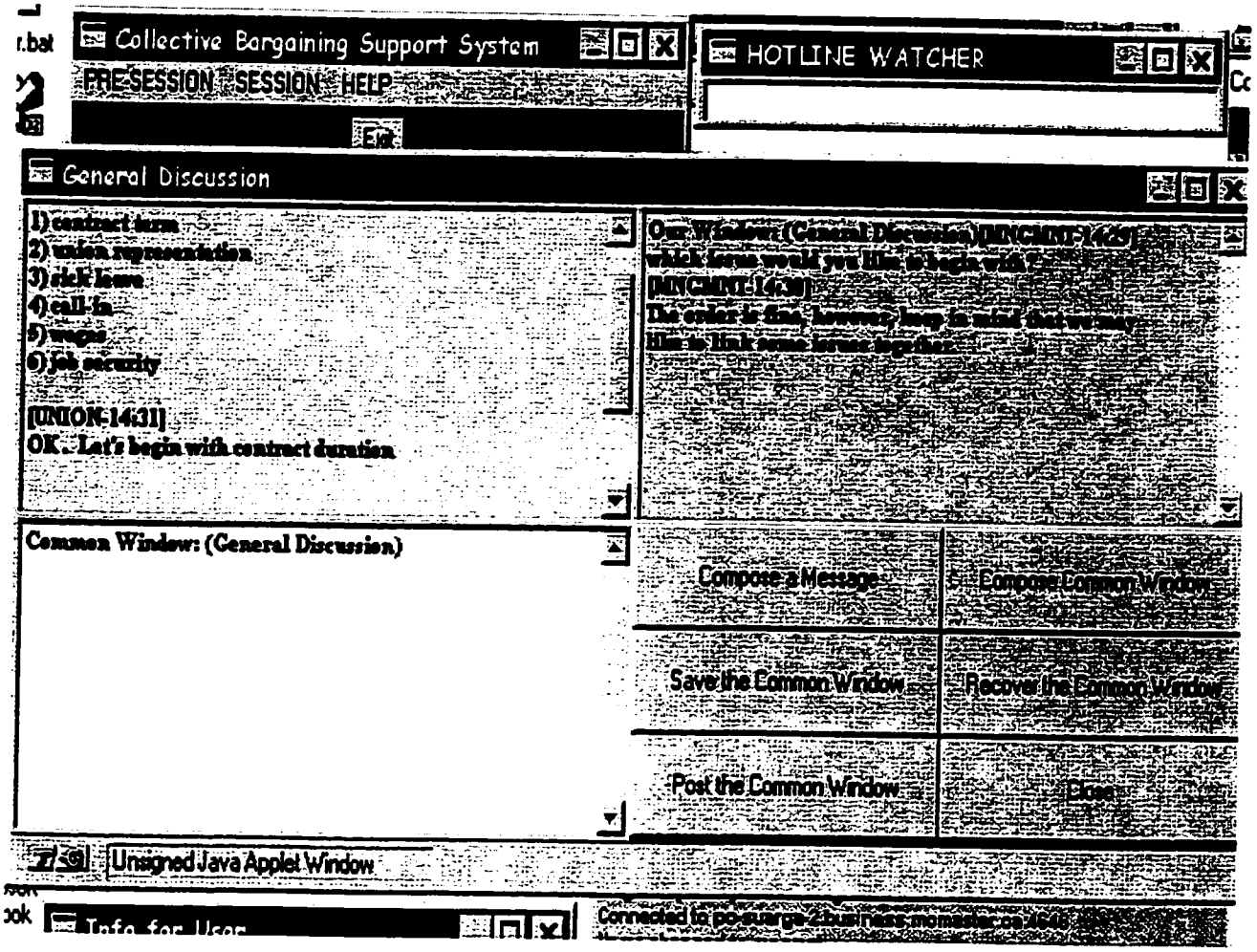


Figure 4.14 - General Discussion Window and Compose a Message button

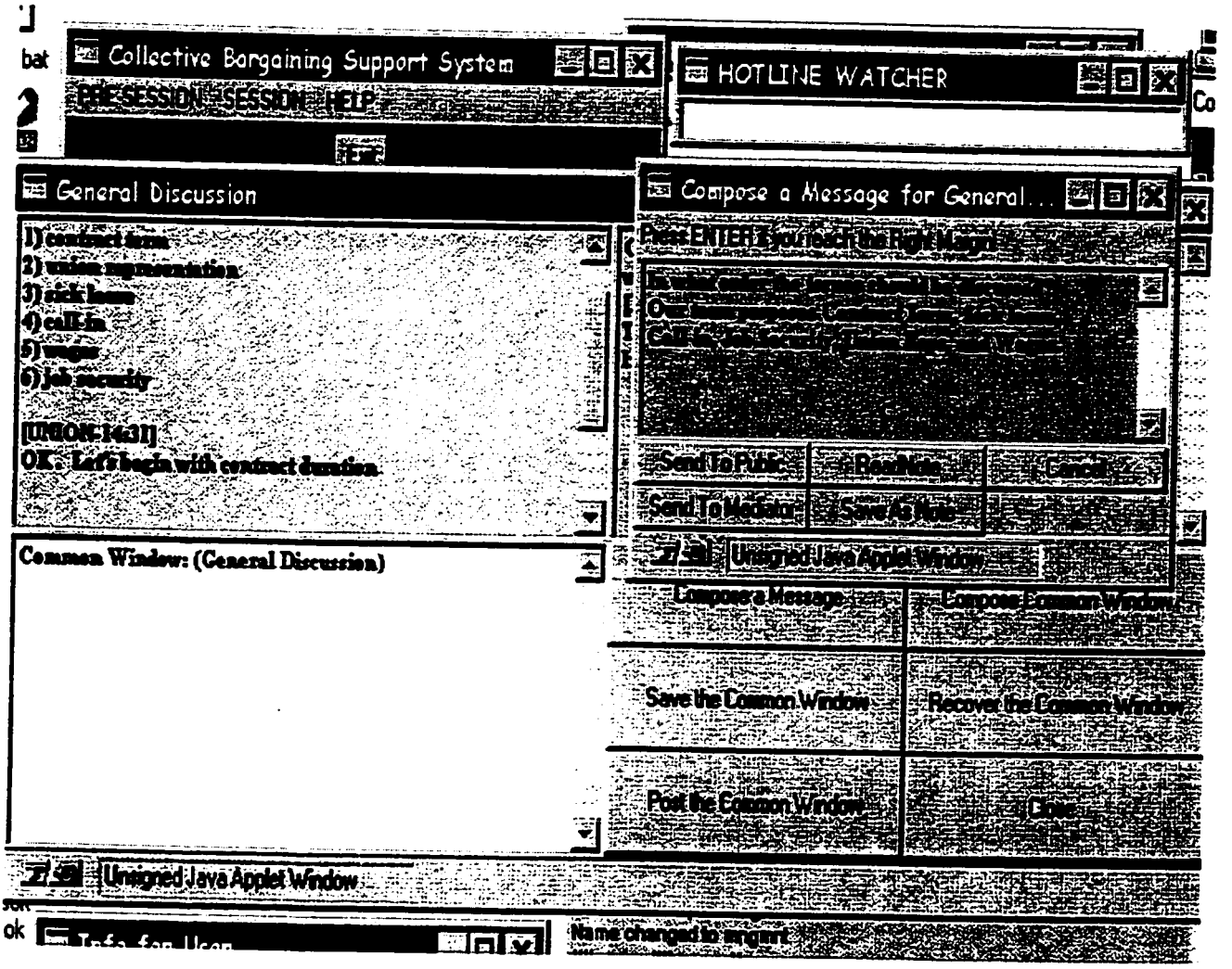


Figure 4.15 - Composing a Message in the Comment Editor

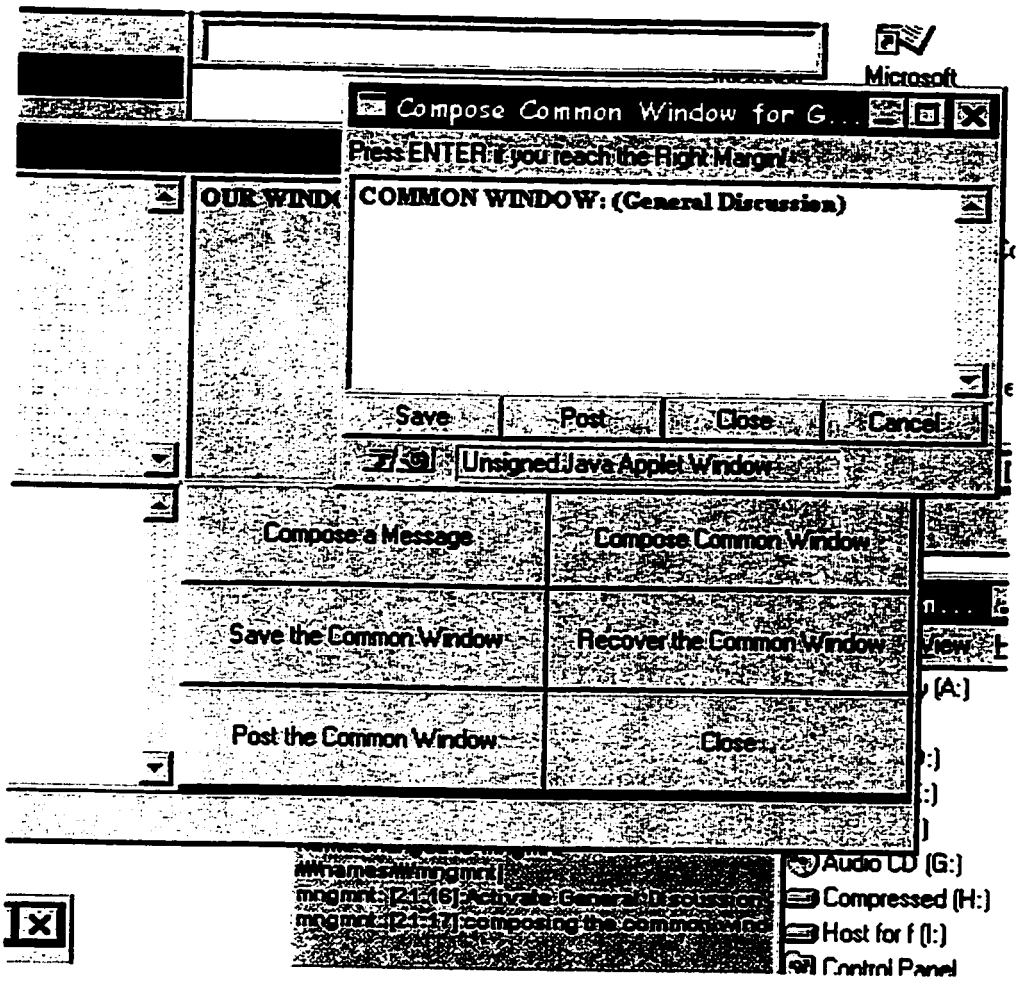


Figure 4.16 Compose Common Window button and the Comment Editor.

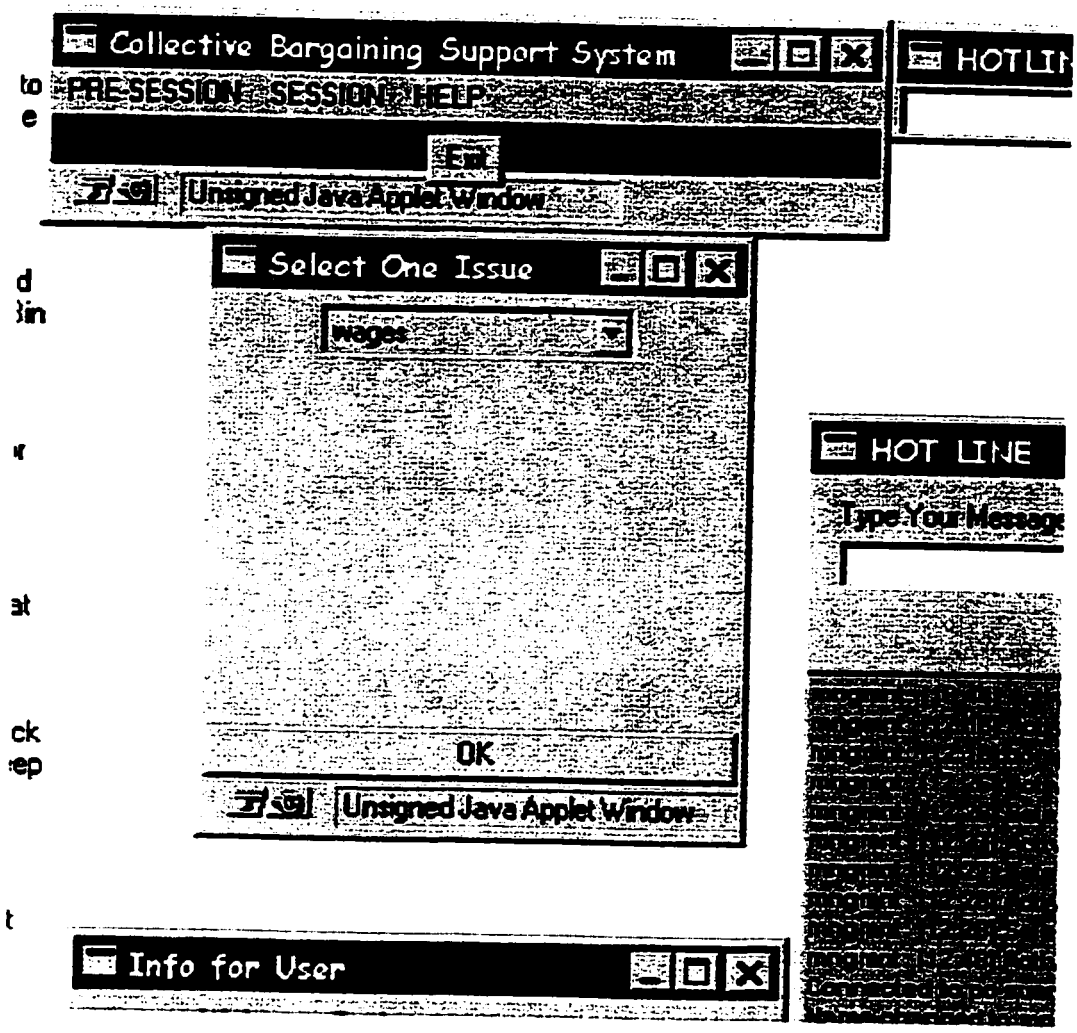


Figure 4.17 - Select One Issue

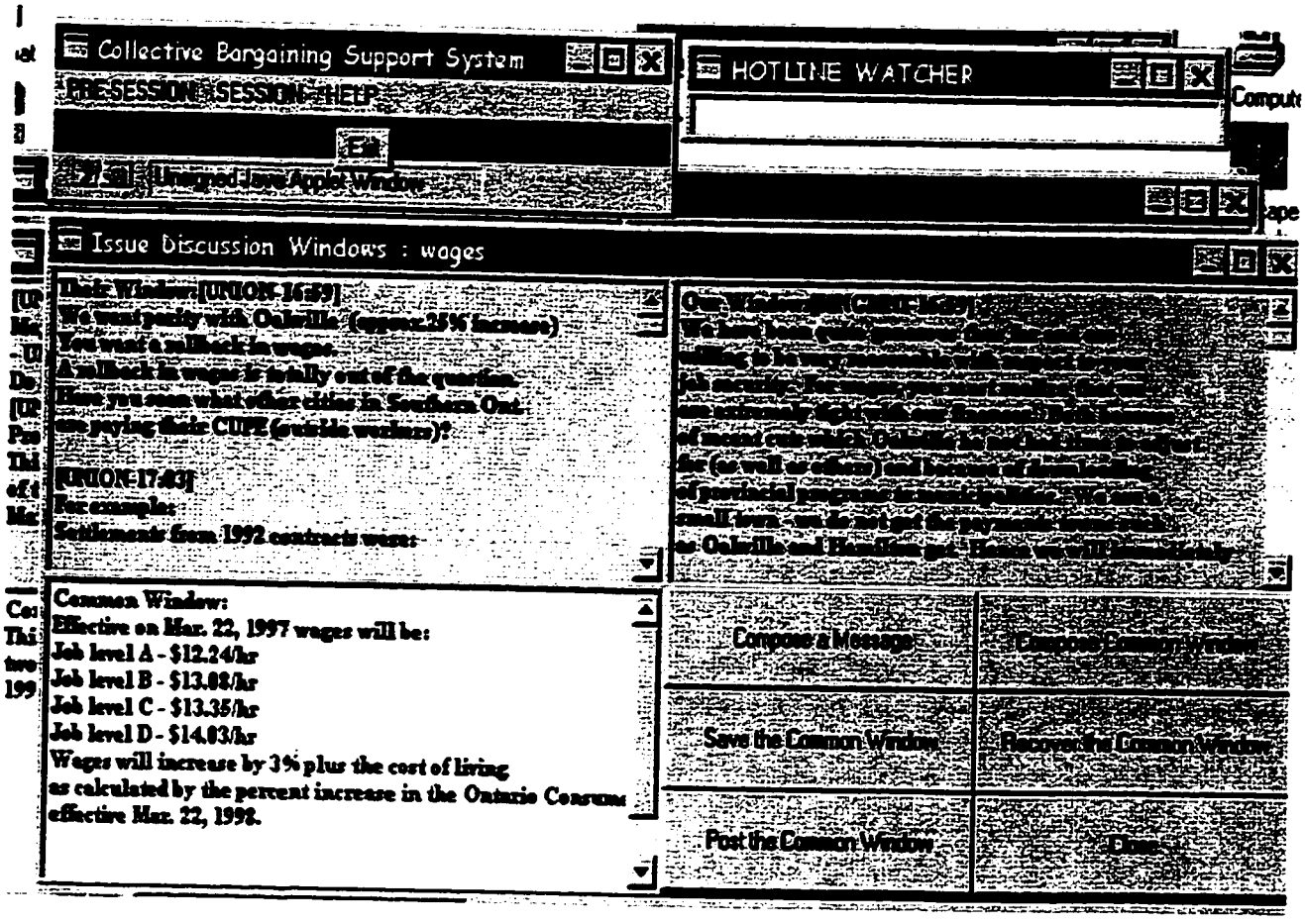


Figure 4.18 - Issue Discussion Window

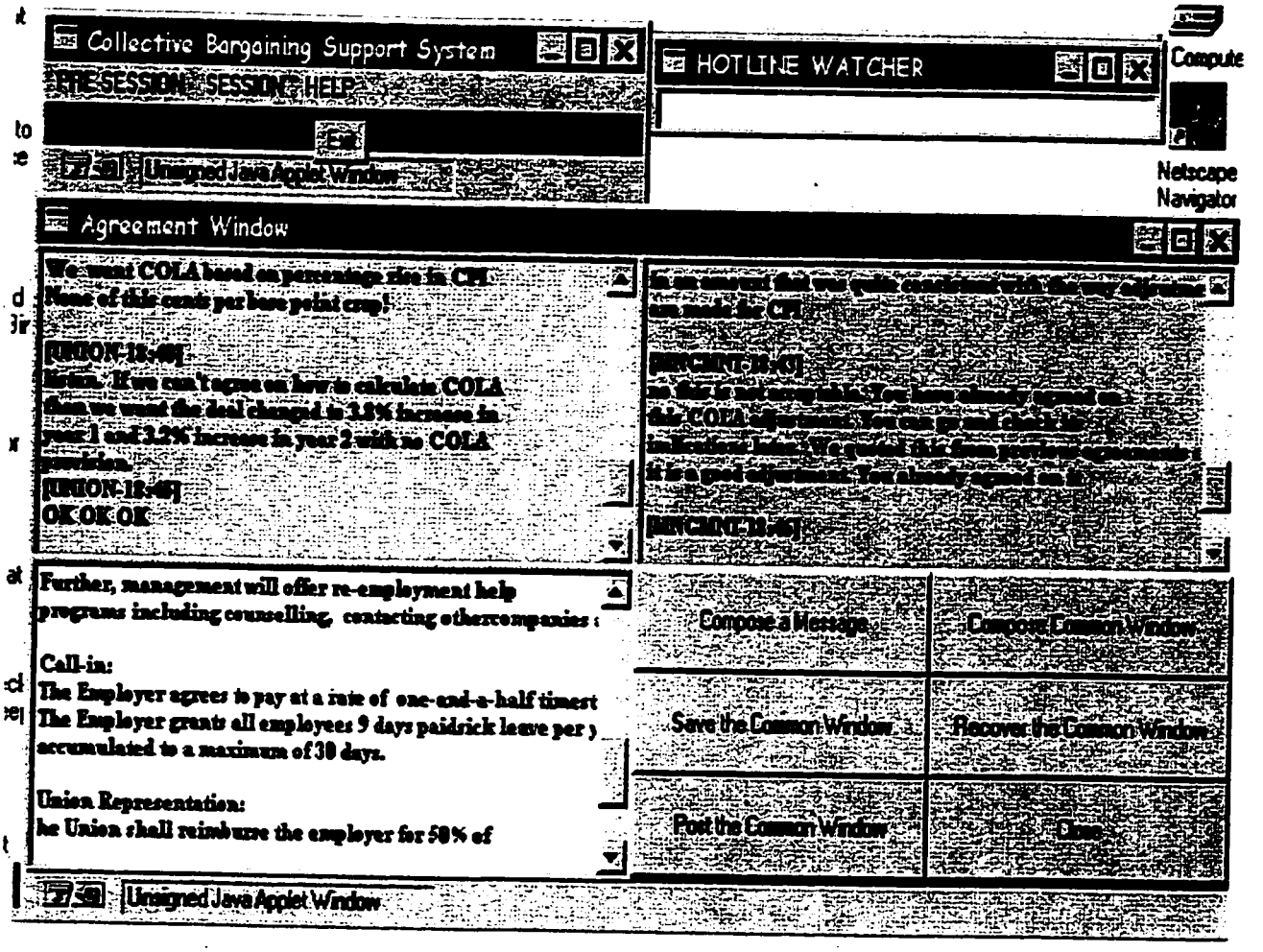


Figure 4.19 - Agreement Window

Chapter 5. Hypotheses, Experiments, and Data Analysis.

5.1 Hypotheses.

Five hypotheses were formulated in this research concerning the validity of CBSS to be used as an alternative in a collective bargaining process, the time efficiency of using CBSS compared to the face-to-face (FTF) process, and the effectiveness of CBSS as a negotiation tool. Whether negotiating with CBSS is a better process than FTF negotiation and whether CBSS affect the negotiation outcomes negatively was also investigated.

Each hypothesis was examined by analyzing responses to several questions from two questionnaires Q1 or Q2. The hypotheses and their corresponding questions are as follows:

Hypothesis 1: Negotiating with CBSS is acceptable as an alternative to FTF negotiation.

This hypothesis was tested based on student responses to the following questions:

$H_{1.1}$: I prefer to use CBSS to negotiate rather than a FTF meeting (Q2-18).
$H_{1.2}$: I prefer to use CBSS combined with a FTF meeting (Q2-19).
$H_{1.3}$: I prefer to use CBSS if FTF meeting is not possible (Q2-20).

These three questions represent the different degree of acceptance from the strongest (unconditional) to weakest (conditional).

Hypothesis 2: Negotiating with CBSS is slower than FTF.

This hypothesis was tested based on student responses to the following questions:

$H_{2.1}$: Process with CBSS was more efficient in time usage than FTF process (Q2-4).
$H_{2.2}$: Process with CBSS was faster than FTF because of this software (Q2-7).

$H_{2.3}$: Compared to FTF, CBSS slowed down the negotiation process (Q2-12).
$H_{2.4}$: CBSS made the process slower than the traditional FTF (Q2-16).

These questions were asked after both methods have been used by each team.

Hypothesis 3: CBSS is effective support tool for negotiation.

This hypothesis was tested based on student responses to the following questions:

$H_{3.1}$: CBSS helps me in preparing the final contract agreement (Q1-5).
$H_{3.2}$: CBSS made valuable contribution to the negotiation outcomes (Q1-6).
$H_{3.3}$: CBSS made the negotiation process easy to organize (Q1-8).
$H_{3.4}$: CBSS is a user-friendly computer software (Q1-12).
$H_{3.5}$: CBSS may be used for real bargaining situations (Q1-13).
$H_{3.6}$: CBSS is the kind of software that I would avoid using (Q1-14).

The first four questions address different benefit the CBSS may provide, and the last two represent the intention of using CBSS.

Hypothesis 4: Negotiating with CBSS is a better process than FTF negotiation.

This hypothesis was tested based on student responses to the following questions:

$H_{4.1}$: CBSS process was more successful than FTF process (Q2-1).
$H_{4.2}$: Process with CBSS was better in attaining my team' s goal than FTF process (Q2-2).
$H_{4.3}$: Process with CBSS gave a more satisfying contract outcome than FTF process (Q2-3).
$H_{4.4}$: Process with CBSS was more effective than FTF process (Q2-5).
$H_{4.5}$: Process with CBSS was easier than FTF because of this software (Q2-6).
$H_{4.6}$: Process with CBSS was better overall than FTF process because of this software (Q2-8).

These questions were designed to compare CBSS with FTF after each team has used both methods.

Hypothesis 5: CBSS does not negatively affect bargaining outcomes.

This hypothesis was tested based on the teaching assistant' s evaluation of student team reports on the final bargaining outcome.

5.2 Experimental Design

The hypotheses were tested by using data collected from respondents who used CBSS in collective bargaining simulations. Three questionnaires were designed for collecting data (see Appendix G). Each questionnaire contained several questions about CBSS usage and a comparison between CBSS and FTF negotiations. Design of these questions was inspired by questions contained in Carmel et.al [1993] and Foroughi et.al [1995], and by the objectives of this research.

The first questionnaire (Q1) contained questions concerning the CBSS as a support tool for negotiations. The second questionnaire (Q2) contained questions regarding a comparison between CBSS and FTF negotiations after both methods were used by each team. These two questionnaires used a Likert scale ranging from 1 to 5, where 1=strongly disagree, 2=disagree, 3=neutral/undecided, 4=agree, and 5=strongly agree. The third questionnaire (Q3) was designed to collect user opinions on CBSS usage as a negotiation support system. This questionnaire contained open ended questions to be answered by CBSS users.

Two experiments were conducted. The first experiment was conducted involving undergraduate students who took collective bargaining courses. This experiment was performed in two rounds. In each round data were collected by distributing the first two questionnaires (Q1 and Q2) respectively. The main purpose of the first experiment is to do the comparison between CBSS and FTF and to test the hypotheses. The second experiment was conducted involving graduate students who took MBA collective bargaining course. This experiment was performed in one round. The second experiment was designed to explore more insights into CBSS by collecting user opinions on CBSS usage, and to verify whether user opinions supported the results of hypothesis tests obtained from the first experiment. Questionnaire Q3 was distributed in the second experiment.

Both CBSS experiments were conducted in six phases. In Phase 1, subjects filled out a consent form. In Phase 2, subjects were given a short training session and an user' s guide. The user' s guide contained easy to follow instructions on how to use CBSS In Phase 3, students were allowed to try the software for 30 minutes in order to familiarize themselves before using CBSS for negotiation. In Phase 4, subjects conducted a Union-Management negotiation for 2.5 hours to reach a contract agreement on the given issues. In Phase 5, subjects completed a questionnaire, and in Phase 6 the students submitted a report to be evaluated concerning their bargaining strategy and their team' s performance in the bargaining process.

5.2.1 The First Experiment.

The first experiment was designed to include three independent variables (the bargaining format, the team' s role, and the bargaining task), each with two treatments (CBSS or face-to-face negotiation (FTF), management or union role, first collective agreement or renewal collective agreement respectively (see Appendix E)). 66 students enrolled in an upper-level undergraduate Collective Bargaining course voluntarily agreed to participate in the first experiment. Some of these students were part-time students who worked in industry as managers or as employees. The students were divided into 22 teams (3 members each) participating in 11 simulated negotiation settings. Each setting involved a management team and a union team. The teams were selected randomly to be in negotiation settings as described below.

In the first round of the first experiment, there were 5 CBSS negotiations and 6 face-to-face (FTF) negotiations. In the second round, 5 involved FTF negotiations and 6 used CBSS. The bargaining formats were switched in the second round, the teams with CBSS simulation in the first round conducted FTF in the second round, and vice versa. The roles of the teams in the simulations were rotated in the second round, so that the teams which were management in the first round became union in the second round, and vice versa.

The collective bargaining simulations were conducted in two isolated rooms (one room for each team) on computers connected to the Internet. Two sets of issues were given to the dyads, one set for the first round, and the other set for the second round. The first round simulated the negotiation of a first collective agreement and included these issues: Wages, Contract duration, Call-in provision, and Union security. The second round involved a renewal collective agreement and covered these issues: Seniority, Layoffs, and Union representation. Case materials were given to the students several weeks prior to conducting the simulations to allow them sufficient time to research the issues (see Appendix E). The experimental design is described in the following figure (Figure 5.1). The research model is a three-factor 2 x 2 x 2 factorial design.

	Format/Method:			
	CBSS		FTF	
Role:	Union	Mngmnt	Union	Mngmnt
Task :				
First Agreement	5 teams	5 teams	6 teams	6 teams
Renewal Agreement	6 teams	6 teams	5 teams	5 teams

Figure 5.1 Experimental design

The students who conducted negotiations with CBSS in the first round were given the first questionnaire (Q1), and in the second round all students were given the second questionnaire (Q2). There were 96 sets of questionnaires distributed. 30 of Q1 were distributed in the first round, and 66 of Q2 were distributed in the second round. The students filled out the questionnaires independently. The number of completed questionnaires returned were 27 sets of questionnaire Q1 and 54 sets of questionnaire Q2.

5.2.2 The Second Experiment.

The second experiment was conducted about four months later, and involved nine full-time and part-time graduate students in an MBA Collective Bargaining course. These students were grouped 2 union teams and 2 management teams. However these teams did not know their role until immediately prior to the start of the simulations. Two negotiation settings were performed during this experiment, and the negotiating teams conducted collective bargaining for 4 hours using CBSS to negotiate six issues (contract term, wages, job security, call-in, sick leave, union representation (see Appendix F)) covering the first collective agreement. There were no FTF simulations in the second experiment. Questionnaire Q3, containing open ended questions, was distributed in order to gather opinions and insights on CBSS usage.

5.3 Data Consistency Test

Data collected from questionnaires during the first experiment were validated before being analyzed. A reliability test was performed in order to assess the internal consistency of the data, i.e., how consistently individuals responded to questions.

The reliability coefficient used in this test was Cronbach' s alpha [Cronbach, 1951] which is a standard test generally used for measuring internal consistency of subjects' responses to questions on Likert scale. A high Cronbach' s alpha value indicates that data are consistent, or respondents were consistent in answering all questions. It is usually accepted that the responses are consistent if Cronbach' s alpha coefficient is above 0.60.

The reliability test was performed by SimStat Ver. 3.5, for questionnaire Q1 and Q2, and the results are as follows:

Questionnaire #1 Cronbach Alpha coefficient = 0.7383
Questionnaire #2 Cronbach Alpha coefficient = 0.7320

Since the Cronbach' s alpha coefficients for both questionnaires were greater than .60 then it can be concluded that the responses to the questions in Q1 and Q2 were consistent.

The reliability test for questions that support each hypothesis were also performed, and the results are as follows:

Hypothesis	Cronbach' s Alpha
H1	0.6385
H2	0.7905
H3	0.6999
H4	0.8694

The results show that the responses to questions that support each hypothesis were reasonably consistent.

5.4 Data Analysis

The hypothesis tests, questionnaire Q2 responses analysis, user comments, and some user opinions from questionnaire Q3 are presented in this section.

5.4.1 Hypothesis Tests.

The first four hypotheses were tested by analysing the data collected from the questionnaires. The scores of the questions that support each hypothesis were collected from every students who participated in the experiments. Hypothesis tests were conducted in two parts. In the first part, questions or sub-hypotheses were examined one by one in order to see how well these questions support the hypothesis. In the second part, each hypothesis is formulated as an aggregation of related sub-hypotheses.

5.4.1.1 Sub-hypothesis Tests.

Each questions was tested using one-tail Wilcoxon Signed Rank test. The null and alternate hypothesis are stated as follows:

Null hypothesis ($H_{oi.j}$) : $M_{i.j} = 3$ where Mij was the estimated sample median of responses to question j of hypothesis i.

Alternate hypothesis ($H_{ai.j}$) : either $M_{ij} > 3$, or $M_{ij} < 3$ depends on the questions.

Hypothesis 1: Negotiating with CBSS is acceptable as an alternative to FTF negotiation.

Hypothesis H1 was examined by three questions which addressed the preference of using CBSS rather than FTF. The statistical analysis results for these individual questions are shown in Table 5.1. Two questions ($H_{1.2}$, $H_{1.3}$) have significant results, the null hypothesis (that the median=3.0) for these questions were rejected, and the alternative hypothesis (that

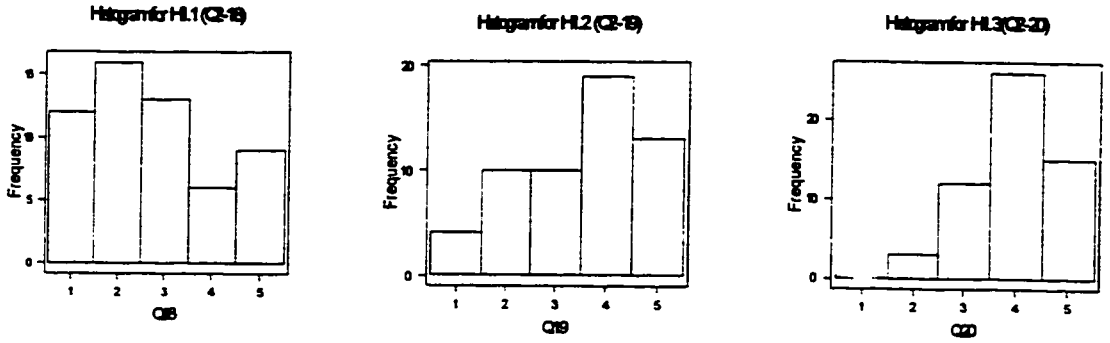
median > 3) could not be rejected. Since the sample median greater than 3, most respondents agreed to these sub-hypotheses. The test result for $H_{1.1}$ was not significant, the null hypothesis (that the median=3) for this question was not rejected. In other words, respondents were neutral to this sub-hypothesis. The responses to $H_{1.1}$ was mixed, some respondents were in favour of FTF negotiations if they have to chose either FTF or CBSS negotiations. Probably because respondents were not familiar yet with this new way to conduct negotiations. However, other respondents preferred to use CBSS rather than FTF. Figure 5.2 shows the histograms of responses to questions $H_{1.1}$, $H_{1.2}$, and $H_{1.3}$.

Table 5.1 Statistical analysis result of questions in hypothesis H1.

Question	N	Median	W	p
$H_{1.1}$: I prefer to use CBSS to negotiate rather than a FTF meeting (Q2-18).	43	2.5	366	0.099(ns)
$H_{1.2}$: I prefer to use CBSS combined with a FTF meeting (Q2-19).	46	3.5	779	0.005(**)
$H_{1.3}$: I prefer to use CBSS if FTF meeting is not possible (Q2-20).	44	4	945	0.000(**)

Note: Median= estimated sample median, W = Wilcoxon statistic, N= n for test
Alternative hypothesis for sub-hypothesis $H_{1.1}$ was median < 3.0.
Alternative hypothesis for sub-hypotheses $H_{1.2}$ and $H_{1.3}$ was median > 3.0.

Our conclusions were that negotiating with CBSS is acceptable as a conditional alternative to a FTF negotiation (as per $H_{1.3}$), and that respondents prefer to use it in combination with FTF (as per $H_{1.2}$). We can conclude that, although CBSS is not unconditionally preferred to FTF, it is preferred in certain circumstances.



$H_{1.1}$ (Q2-18) $H_{1.2}$ (Q2-19) $H_{1.3}$ (Q2-20)

Figure 5.2 Hypothesis H1 histograms

Hypothesis 2: Negotiating with CBSS is slower than FTF.

Hypothesis H2 is supported by four sub-hypotheses: the process with CBSS was more efficient in time usage than FTF process ($H_{2.1}$), the process with CBSS was faster than FTF ($H_{2.2}$), CBSS slowed down the negotiation process ($H_{2.3}$), and CBSS made the process slower than the traditional FTF ($H_{2.4}$). The first two questions are in opposite directions with the other two questions. However, principally these sub-hypotheses measure the same thing, whether negotiating with CBSS was less efficient than FTF negotiation. Statistical analysis results of these similar questions proved that responses were consistent.

Statistical analysis results for each question are shown in Table 5.2. All test results for these sub-hypotheses were significant, the null hypothesis for these questions was rejected, which means that most respondents disagreed that CBSS was more efficient in time usage than FTF, disagreed that CBSS was faster, agreed that CBSS slowed down the negotiation

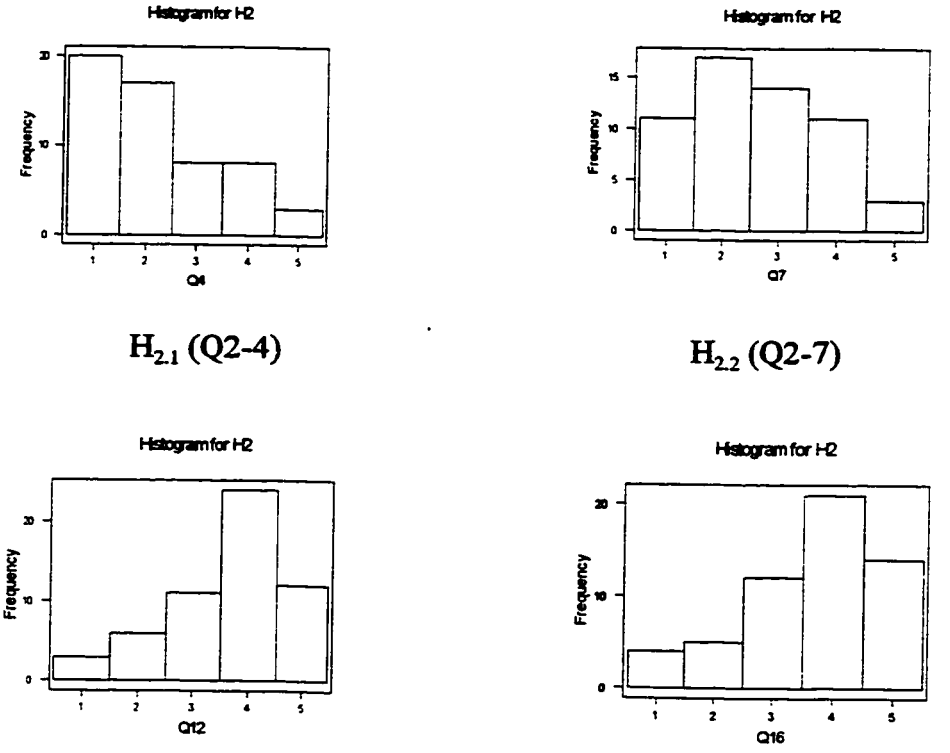
process, and agreed that CBSS made the process slower than the traditional FTF. Our conclusion, hypothesis H2 should not be rejected. In other words, most respondents agreed that negotiating with CBSS is slower than FTF. This result confirms the result reported by Foroughi et.al [1995] that "...the negotiation time was greater for NSS dyads than for non NSS dyads...". The conditional acceptance of CBSS as an alternative to FTF negotiation was probably because respondents perceived CBSS negotiation was slower than FTF negotiation.

Table 5.2 Statistical analysis results of questions in hypothesis H2

Question	N	Median	W	p
$H_{2.1}$:Process with CBSS was more efficient in time usage than FTF process (Q2-4).	48	2	215	0.000(**)
$H_{2.2}$:Process with CBSS was faster than FTF because of this software (Q2-7).	42	2.5	266	0.010(**)
$H_{2.3}$:Compared to FTF, CBSS slowed down the negotiation process (Q2-12).	45	3.75	828	0.000(**)
$H_{2.4}$:CBSS made the process slower than the traditional FTF (Q2-16).	44	3.75	780	0.000(**)

Note: alternative hypothesis for $H_{2.1}$ and $H_{2.2}$ was median < 3,
alternative hypothesis for $H_{2.3}$ and $H_{2.4}$ was median > 3.

There are some reasons why negotiators may perceive that CBSS slows down the negotiation. Negotiators perceived delays in receiving responses from other parties during the bargaining simulations using CBSS. The time involved in the processes supported by CBSS are: the time for keying in the message, the time for waiting the response from the other side, and the time to conduct informal caucus meetings, which added up to more than face-to-face processes. The histogram of responses of questions in hypothesis H2 is shown in Figure 5.3.



$H_{2.1}$ (Q2-4)

$H_{2.2}$ (Q2-7)

$H_{2.3}$ (Q2-12)

$H_{2.4}$ (Q2-16)

Figure 5.3 Hypothesis H2 histograms

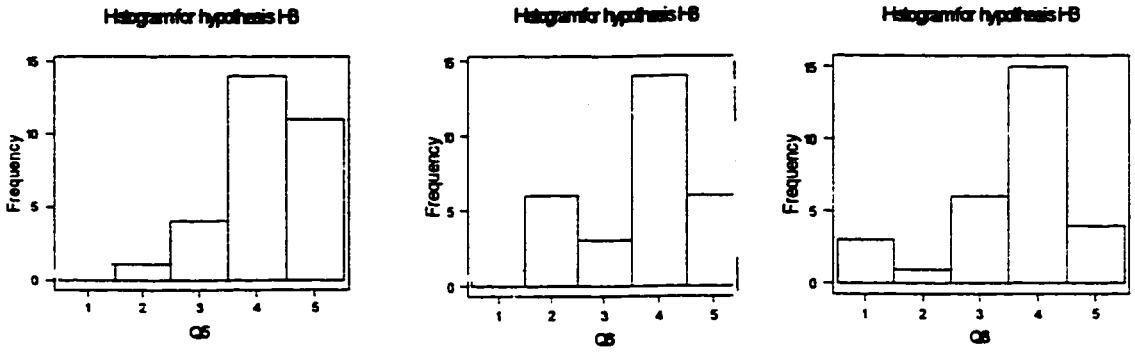
Hypothesis 3: CBSS is an effective support tool for negotiation

Hypothesis H3 was covered by six questions listed in Table 5.3. The statistical analysis shows significant results in each case, which means that null hypothesis (that median=3) must be rejected. Most respondents agreed that CBSS was an effective tool because CBSS helped them in preparing the final contract agreement ($H_{3.1}$), and CBSS made valuable contributions to the negotiation outcomes ($H_{3.2}$). Most respondents also agreed that CBSS made the negotiation process easy to organize ($H_{3.3}$), it is user friendly software ($H_{3.4}$), and it may be

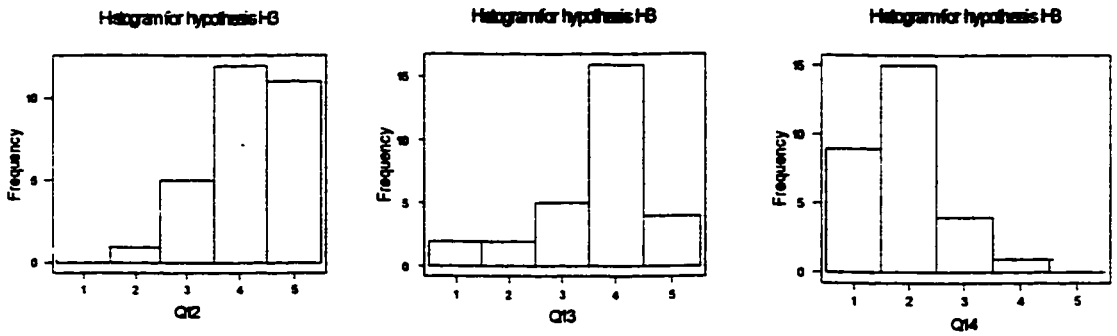
used for real bargaining situations ($H_{3.5}$). As an acceptable support tool, CBSS would not be avoided by respondents. In other words, most respondents disagreed that CBSS is the kind of software that they would avoid using ($H_{3.6}$). Our conclusion, hypothesis H3 that CBSS is an effective support tool for negotiation was not rejected. In other words, CBSS is an effective support tool for negotiation. Histograms of responses to questions in hypothesis H3 are shown in Figure 5.4.

Table 5.3 Statistical analysis results of questions in hypothesis H3

Question	N	Median	W	p
$H_{3.1}$: CBSS helped me in preparing the final contract agreement (Q1-5).	25	4.5	325	0.000(**)
$H_{3.2}$: CBSS made valuable contributions to the negotiation outcomes (Q1-6).	26	4	288	0.002(**)
$H_{3.3}$:CBSS made the negotiation process easy to organize (Q1-8).	23	3.5	207.5	0.018 (*)
$H_{3.4}$:CBSS is a user-friendly computer software (Q1-12)	24	4	293	0.000(**)
$H_{3.5}$:CBSS may be used for real bargaining situations(Q1-13).	24	4	238	0.004(**)
$H_{3.6}$: CBSS is the kind of software that I avoid using (Q1-14).	25	2	8.5	0.000(**)



$H_{3.1}$ (Q1-5) $H_{3.2}$ (Q1-6) $H_{3.3}$ (Q1-8)



$H_{3.4}$ (Q1-12) $H_{3.5}$ (Q1-13) $H_{3.6}$ (Q1-14)

Figure 5.4 Hypothesis H3 histograms

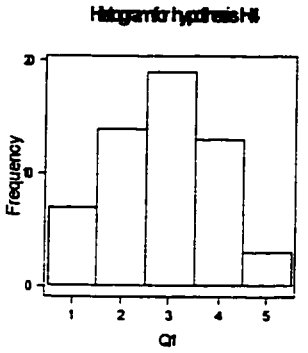
Hypothesis 4: Negotiating with CBSS is a better process than FTF negotiation.

This hypothesis was examined with six questions: the bargaining process with CBSS was more successful ($H_{4.1}$), was better in attaining goals ($H_{4.2}$), gave more satisfying contract ($H_{4.3}$), was more effective ($H_{4.4}$), was easier ($H_{4.5}$), and was better overall ($H_{4.6}$) than FTF meetings.

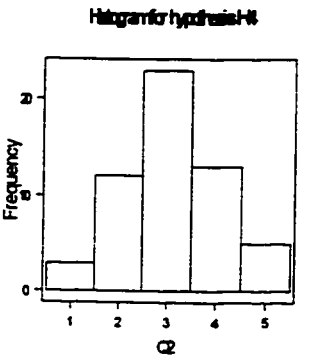
As we can see in Table 5.4, statistical analysis results of these questions are not significant, the null hypothesis that median=3 could not be rejected, which mean that most respondents were neutral to these questions. The estimated sample medians were 3 (neutral) except for sub-hypothesis $H_{4.4}$ which was 3.50. In this case, most respondents were in favour of CBSS, they believe that process with CBSS was more effective than FTF process. The conclusion, hypothesis H4 that negotiating with CBSS is better process than FTF negotiation was rejected. Probably, it can be concluded that process with CBSS is as good as an FTF meeting. The histograms of responses to questions in hypothesis H4 are shown in Figure 5.5.

Table 5.4 Statistical analysis result of questions in hypothesis H4

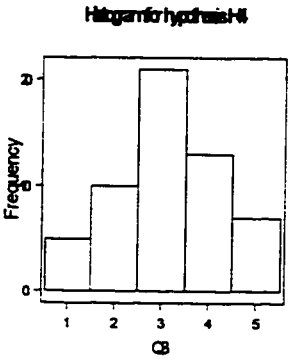
Question	N	Median	W	p
$H_{4.1}$: CBSS process was more successful than FTF process (Q2-1).	37	3	279	0.863(ns)
$H_{4.2}$: Pocess with CBSS was better in attaining my team' s goal than FTF process (Q2-2).	33	3	316.5	0.283(ns)
$H_{4.3}$: Process with CBSS gave me a more satisfying contract outcomes than FTF process (Q2-3).	35	3	362.5	0.221(ns)
$H_{4.4}$: Process with CBSS was more effective than FTF process (Q2-5).	43	3.5	593	0.075(ns)
$H_{4.5}$: Process with CBSS was easier because of this software (Q2-6).	44	3	462	0.652(ns)
$H_{4.6}$: Process with CBSS was better overall because of this software (Q2-8).	39	3	334	0.785(ns)



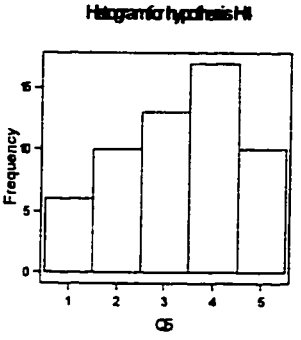
$H_{4.1}$ (Q2-1)



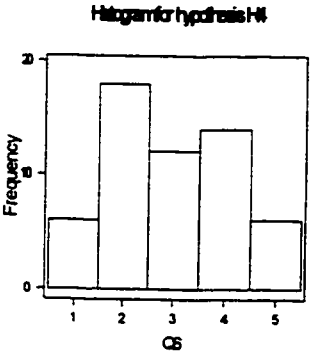
$H_{4.2}$ (Q2-2)



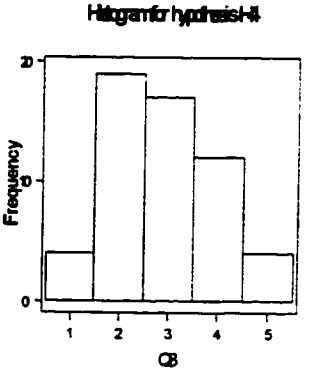
$H_{4.3}$ (Q2-3)



$H_{4.4}$ (Q2-5)



$H_{4.5}$ (Q2-6)



$H_{4.6}$ (Q2-8)

Figure 5.5 Hypothesis H4 histograms

Histograms in Figure 5.5 show that responses to the questions in hypothesis H4 were mixed. Some respondents were neutral, some disagreed, and others were agreed to the questions. So, in spite of aggregated result which was neutral, some respondents believed that negotiation with CBSS was a better process than FTF negotiation, but some others disagree.

Hypothesis 5: CBSS does not negatively affect bargaining outcome.

Hypothesis H5 was tested by analyzing the grades that student teams received on their reports in the first experiment. As mentioned in Section 5.2, student teams were required to submit a report in Phase 6 of the experiment. The reports discussed bargaining strategy, initial demands (initial offers to the opposite party) and resistance points (the worst but still acceptable offers from the opposite party) on the issues, and the final results. The bargaining outcomes were evaluated relative to the resistance points and initial demands, and grades were assigned to the reports by a teaching assistant accordingly. The grading system assigned by the teaching assistant for each issue was as follows:

- if the final outcome was worse than the resistance point, the grade = 0,
- if the final outcome was better than resistance point but worse than the initial demand, the grade = .10,
- if the final outcome equals the initial demand, the grade = .15.

The grade average of all issues in a negotiation setting was used as the report' s grade. The report' s grading system was re-scaled so that the minimum grade was 0 and the maximum grade was 5. These grades are tabulated in Table 5.8 which shows that there are three factors attached to the grade: bargaining method (CBSS or FTF), the role (Management or Union), and the task (first round or second round). In order to test hypothesis H5, a non-parametric two-sample test (Wilcoxon rank sum test) was performed. The test shows whether the median of report scores of the teams who conducted negotiation with CBSS was

equal to the median of report scores of the team who conducted FTF negotiation, the result is shown in Table 5.5.

Table 5.5 Two-sample Wilcoxon rank sum test of CBSS teams versus FTF teams

CBSS-Median	FTF-Median	N	MW	p-value	Conclusion
3.33	3.75	22	434	0.15(ns)	do not reject H5 : the median scores are not significantly different.

The result in Table 5.5 shows that the null hypothesis that the median of team reports with CBSS is equal to the median of the team reports with FTF was not rejected. In other words, the format of the negotiations method, whether the teams used CBSS or FTF, did not have a significant effect on the teams' final grade. In conclusion, the bargaining method did not affect the bargaining outcome evaluations. That is, statistical analysis results support hypothesis H_5 that CBSS does not have negative effects on the bargaining outcome evaluations. (Note: among 11 negotiation sessions only one negotiation failed to settle. Therefore this conclusion is in terms of the 91% of settlements actually reached).

5.4.1.2 Integrated Hypothesis Tests.

In the second part of hypothesis tests, each hypothesis was formulated as a function of related questions. The test procedure is described as as follows:

(1) Formulate each hypothesis as a function of questions and assume that each question has a same weight. Response to hypothesis H_i is defined as :

$$Y_1 = (M_{11} - 3) + (M_{12} - 3) + (M_{13} - 3);$$

$$Y_2 = (3 - M_{21}) + (3 - M_{22}) + (M23 - 3) + (M_{24} - 3);$$

$$Y_3 = (M_{31} - 3) + (M_{32} - 3) + (M_{33} - 3) + (M_{34} - 3) + (M_{35} - 3) + (3 - M_{36});$$

$$Y_4 = (M_{41} - 3) + (M_{42} - 3) + (M_{43} - 3) + (M_{44} - 3) + (M_{45} - 3) + (M_{46} - 3);$$

Where M_{ij} = response to question j of hypothesis H_i

Here, 3 is subtracted from each measure since this is the neutral response in each case.

(2) Null hypothesis : sample median $Y_i \leq 0$; in this case do not reject hypothesis H_i.

(3) Alternative hypothesis : sample median of $Yi > 0$; reject hypothesis H_i.

The statistics test was performed by using Minitab Release 11 for Windows (one sided Wilcoxon Signed Rank test) with 95% level of confidence. Responses to the questions which have opposite direction compared to other questions (e.g., $H_{2.1}$ and $H_{2.2}$) were reversed accordingly. The results are shown in the following tables (Table 5.6).

Table 5.6 Statistical test result of hypotheses H1 to H4

Hypothesis	N	Median (Yi)	W-value	p
H1	53	1	1087.5	0.001 (**)
H2	52	2.5	1130.5	0.000 (**)
H3	29	3.5	379	0.000 (**)
H4	53	0	723	0.475(ns)

(*) significant at the 0.05 level; (**) significant at the 0.01 level;
ns = not significant.

As we can see in Table 5.6, the statistical test result for hypothesis H1 is very significant (p=0.001). In other words, null hypothesis for H1 (median $\leq$ 0) was rejected. This result confirmed the conclusion in the first part of the hypothesis tests, that negotiating with CBSS is acceptable as a conditional alternative to FTF negotiation.

The statistical analysis result for H2 shown in Table 5.1 is very significant (p=0.000) which means that null hypothesis for H2 (median $\leq$ 0) was rejected. This result supported the conclusion in the first part that negotiating with CBSS was slower than FTF negotiation.

The statistical test of hypothesis H3 shown in Table 5.1 was very significant (p=0.000) which means that null hypothesis H3 (median $\leq$ 0) was rejected. As a consequence, CBSS is acceptable as an effective support tool for negotiation.

The statistical analysis of the result for H4 shown in Table 5.1 is not significant (p=0.475) which means that null hypothesis for H4 (median $\leq$ 0) was not rejected, or respondents were neutral to hypothesis H4. This result support the conclusion of data analysis in the first part (section 5.4.1.1) which also concluded that negotiating with CBSS was as good as FTF negotiations.

5.4.2 Comparing the CBSS-first group with FTF-first group.

As we mentioned before, the first experiment was conducted in two rounds, and in the second round all student participants were given questionnaire Q2. Students can be divided into two groups according to the order of the bargaining method they used in the experiment. The group of students who used CBSS in the first round and then FTF in the second round was named the CBSS-first group (group 1). The group of students who conducted FTF negotiation in the first round and used CBSS in the second round was named the FTF-first group (group 2). In this section, we investigate the factors that affect the students' responses to questionnaire Q2. The statistical results of students' responses to questions in questionnaire Q2, grouped according to the order of method used, are shown in Table 5.7A and Table 5.7B.

According to the statistical analysis in Table 5.7A and Table 5.7B, both groups agreed to statements concerning some characteristics of CBSS, which are: the separation of personalities from the issue problem (question #14), let the users think before replying (question #15), and the process structuring made the negotiation easier (question #17). These characteristics, which represent potential advantages of CBSS over FTF negotiations, are important factors in achieving effective negotiation outcomes.

Allowing personalities to interfere in negotiations can have negative consequences, such as creating misperceptions and emotion, which affect negotiation outcomes. These negative impacts can be reduced by helping the negotiators to focus their attention more on

problems rather than on personalities. In an FTF meeting, negotiators sometimes allow personal matters to affect their judgements. CBSS helps to avoid this by separating the personalities from the issues.

In an FTF meeting a negotiator can be interrupted while delivering an argument, and a negotiator may respond immediately to an argument without thinking because of emotion. CBSS guaranties that negotiators can express their ideas without being interrupted, and provides the opportunity to think before replying to the other side' s arguments.

CBSS also structures the process such that negotiators may move from one issue discussion to another issue discussion without losing any messages. The negotiators may go back to review previous arguments and still able to receive new incoming arguments.

The statistical results presented in Table 5.7A and Table 5.7B also show that on several points (questions) the students of group-1 (CBSS-FTF) and the students of group-2 (FTF-CBSS) reached different conclusions about CBSS. The last column of Table 5.7B shows whether conclusions were different (Diff=Y) or not (Diff=N). There were 10 questions with different conclusions and 10 questions with same conclusions. For example, for question #1, whether negotiating with CBSS was more successful than FTF, CBSS-FTF group had no preference while FTF-CBSS group disagreed. To explain the reasons for these differences, we examined the factors that affect the group characteristics and their effects on the negotiation outcomes.

Three factors that may differentiate the groups and affect the result of the experiment include: (1) whether CBSS was used in the first or second round, (2) the team rotation and (3) the issues that were negotiated. The negotiation methods were changed in the second round, so the dyads with CBSS in the first round became FTF negotiators in the second round, and vice versa. The role of the teams is either Union or Management, and the dyad roles in the first round are reversed in the second round. The team rotation from Union to Management and vice versa may have effects on the results. The task given to the dyads in the first round was first collective bargaining with four issues: Wages, Contract Term, Call-in Provision, and Union Security. The task given to the dyads in the second round was renewal collective bargaining with three issues: Seniority, Layoffs, and Union Representation.

The team report grades in the first round and second round were tabulated (Table 5.8) and analyzed by using three-way non-orthogonal ANOVA [Appelbaum and Cramer, 1974]. The ANOVA results show that the three-factor interaction (method, task, and role) was not significant ($F=0.41$, $n1=1$, $n2=37$, $p>0.05$). However, the two-way interaction test shows that the interaction of task with the role indeed affected the outcomes of the bargaining process ($F=8.75^{**}$, $n1=1$, $n2=37$, $p < 0.01$). The other two-way interactions (task * method and role*method) were not significant. Neither was the main effect for method.

Now, the differences on the questionnaire conclusions can be explained. It seems that role reversal and the nature of the tasks given to the teams in the second round affect the satisfaction of the teams with the outcomes. In other words, by reversing the role of the teams from union to management and vice versa, and at the same time gave them a different

task somehow stimulated dissatisfaction feeling of users toward the negotiation method used in the second experiment. As a consequence, the first group (use CBSS before FTF) rated CBSS more favourably than the the second group (FTF before CBSS). Consider, for example question #3: the process with CBSS was better in attaining my team' s goal than the FTF process. The first group agreed with this statement while the second group had no preference. Consider as well, question #4: the process with CBSS was more efficient in time usage than FTF. The first group had no preference, while the second group which was in favour of FTF disagreed with the statement. To eliminate these side-effects, we may need to conduct more experiments that do not change the role and the task at the same time.

Table 5.7A : Statistical test results of questions in Questionnaire #2 of Group-1 and Group-2

	CBSS-FTF		FTF-CBSS	
Question:	median	p	median	p
#1: CBSS process was more successful than FTF process.	3	0.217	2.5	0.011*
#2: Pocess with CBSS was better in attaining my team' s goal than FTF process.	3.5	0.081	3	0.699
#3: Process with CBSS gave me a more satisfying contract outcomes than FTF process.	3.5	0.037*	3	0.772
#4: Process with CBSS was more efficient in time usage than FTF process.	2.5	0.138	2	0.000**
#5: Process with CBSS was more effective than FTF process.	4	0.010*	3	0.601
#6: Process with CBSS was easier because of this software	3.5	0.143	2.5	0.120
#7: Process with CBSS was faster than FTF because of this software	3	0.793	2.5	0.012*
#8: Process with CBSS was better overall because of this software.	3	0.219	2.5	0.059
#9: Bargaining with CBSS was more relaxed than FTF.	4	0.007**	3	0.505
#10: Bargaining with CBSS was more stressful than FTF.	2.5	0.062	3	0.439
#11: Compare to FTF, the electronic communication support of CBSS improved the communication between the parties.	3	0.489	2.5	0.052
#12: Compare to FTF, CBSS slowed down the negotiation process.	3.5	0.016*	4	0.003**
#13: Compare to FTF, CBSS blocked the communication between the parties.	2.5	0.051	3.5	0.100
#14: Compare to FTF, CBSS helped me to focus more on the issues than on the personalities of the other party members.	4	0.006*	4	0.000**

Table 5.7A : Statistical test results of questions in Questionnaire #2 of Group-1 and Group-2

	CBSS-FTF		FTF-CBSS	
Question:	median	p	median	p
#15: Compare to FTF, CBSS let me think before answering or replying to other side's comments.	4.50	0.000**	4.00	0.000**
#16: CBSS made the process slower than traditional FTF.	3.50	0.047*	4.00	0.002**
#17: The structured of CBSS made the negotiation easier than FTF.	3.50	0.001**	3.50	0.019*
#18: I prefer to use CBSS to negotiate rather than a FTF meeting.	3.50	0.122	2.00	0.002**
#19: I prefer to use CBSS combined with a FTF meeting.	3.50	0.042*	3.50	0.023*
#20: I prefer to use CBSS if FTF meeting is not possible.	4.00	0.001**	4.00	0.000**

Note : Test were performed as one-tail test against 3 (neutral)

Table 5.7B : Conclusions from questions in Questionnaire #2 (Group 1 and Group 2)

	Group - 1			Group - 2			
Question:	Ag	N	DA	Ag	N	DA	Diff
#1: CBSS process was more successful than FTF process.		X				X	Y
#2: Pocess with CBSS was better in attaining my team' s goal than FTF process.		X			X		N
#3: Process with CBSS gave me a more satisfying contract outcomes than FTF process.	X				X		Y
#4: Process with CBSS was more efficient in time usage than FTF process.		X				X	Y
#5: Process with CBSS was more effective than FTF process.	X				X		Y
#6: Process with CBSS was easier because of this software		X			X		N
#7: Process with CBSS was faster than FTF because of this software		X				X	Y
#8: Process with CBSS was better overall because of this software.		X				X	Y
#9: Bargaining with CBSS was more relaxed than FTF.	X				X		Y
#10: Bargaining with CBSS was more stressful than FTF.		X			X		N
#11: Compare to FTF, the electronic communication support of CBSS improved the communication between the parties.		X				X	Y
#12: Compare to FTF, CBSS slowed down the negotiation process.	X			X			N
#13: Compare to FTF, CBSS blocked the communication between the parties.			X		X		Y
#14: Compare to FTF, CBSS helped me to focus more on the issues than on the personalities of the other party members.	X			X			N

Note: Ag = agree, N = neutral, DA = disagree, Diff = any difference? (Y=yes, N=no).

Table 5.7B : Conclusions from questions in Questionnaire #2 (Group 1 and Group 2)

	Group - 1			Group - 2			
Question:	Ag	N	DA	Ag	N	DA	Diff
#15: Compare to FTF, CBSS let me think before answering or replying to other side' s comments.	X			X			N
#16: CBSS made the process slower than traditional FTF.	X			X			N
#17: The structured of CBSS made the negotiation easier than FTF.	X			X			N
#18: I prefer to use CBSS to negotiate rather than a FTF meeting.		X				X	Y
#19: I prefer to use CBSS combined with a FTF meeting.	X			X			N
#20: I prefer to use CBSS if FTF meeting is not possible.	X			X			N

Note: Ag = agree, N = neutral, DA = disagree, Diff = any difference? (Y=yes, N=no).

Table 5.8 : Report evaluation and its factors

Experiment : Round-1, Task=First Collective Agreement (4 issues).

Method: CBSS						Method : FTF					
Team	Role	Grade	Opp. Team	Role	Grade	Team	Role	Grade	Opp. Team	Role	Grade
2	manag	3.75	16	union	2.92	6	union	1.67	11	manag	2.92
8	manag	2.08	21	union	2.08	7	union	4.17	17	manag	2.08
12	manag	3.75	22	union	3.75	9	union	3.75	15	manag	3.75
3	manag	2.5	4	union	2.92	1	union	2.5	10	manag	3.75
5	manag	3.33	19	union	3.33	13	union	3.88	18	manag	4.44
						14	union	2.92	20	manag	3.75

Experiment : Round-2, Task=Renewal Collective Agreement (3 issues).

Method: FTF						Method : CBSS					
Team	Role	Grade	Opp. Team	Role	Grade	Team	Role	Grade	Opp. Team	Role	Grade
2	union	3.89	16	manag	2.22	6	manag	3.33	11	union	3.33
8	union	4.44	21	manag	2.22	7	manag	3.33	17	union	4.44
12	union	3.89	22	manag	3.33	9	manag	1.11	15	union	3.89
3	union	5	4	manag	3.33	1	manag	1.11	10	union	1.11
5	union	3.89	19	manag	2.22	13	manag	2.78	18	union	3.88
						14	manag	2.22	20	union	3.89

Table 5.9 : Statistical test results of questions in Questionnaire Q1

Question:	Median	N	W	p	Concl.
#1:CBSS helps me to focus more on the issue problem rather than on the negotiators' personality	3.50	21	182.5	0.010*	Agree
#2:CBSS helps me in reducing my hostility to the other side' s negotiators	3.00	25	173.0	0.394	No pref
#3:CBSS gave me more time to think before I contributed my ideas into the discussion.	4.50	29	376.0	0.000**	Agree
#4:CBSS helps me in understanding the other side' s negotiators concern and problems.	3.00	17	83.5	0.379	No pref
#5:CBSS helps me in preparing the final contract agreement.	4.50	25	325.0	0.000**	Agree
#6:CBSS had valuable contributions to the negotiation outcomes.	4.00	26	288.0	0.002**	Agree
#7:CBSS sped up the negotiation process.	3.00	25	135.5	0.770	No pref
#8:CBSS made the negotiation process easy to organize	3.50	23	207.5	0.018*	Agree
#9:CBSS improved communication between the two sides.	3.00	17	80.0	0.444	No pref
#10:CBSS reduces my ability to express my ideas clearly.	3.00	25	160.0	0.532	No pref
#11:CBSS is a useful tool for negotiators.	4.00	22	253.0	0.000**	Agree
#12:CBSS is a user-friendly computer software.	4.00	24	293.0	0.000**	Agree
#13:CBSS may be used for real bargaining situations.	4.00	24	238.0	0.006**	Agree
#14:CBSS is a kind of software that I avoid using.	2.00	25	8.5	0.000**	Disagree

Note: the test was performed as one-tail Wilcoxon signed rank test against 3 (neutral).

5.4.5 User Comments

The questionnaires also asked respondents to provide additional comments on the CBSS after using the system. There were 96 sheets of questionnaires distributed to the students (30 sheets with questionnaire Q1 and 66 sheets with questionnaire Q2). 12 sheets were not returned, and 16 sheets were returned without comments. The comments were grouped into four categories: satisfied, frustrated, technical, and mixed comments. Forty one responded with satisfied comments, fourteen with frustrated comments, four commented on the technical difficulties encountered during the experiment, and four had mixed feeling comments. Table 5.10 shows the categories of the comments and their quoted examples.

A good NSS must help users in reducing interpersonal conflicts and support the users in resolving issue-based conflict. The CBSS users comment were analyzed and it was found that most of the satisfied students appreciated the fact that: CBSS separates people from the problem, CBSS gives them the chance to conduct group discussions without being watched by the opposing group, CBSS relieves them from intimidation, CBSS prevents personality interference, CBSS gives them time to think before replying, and CBSS retains the discussion documents such that they can go back to the previous arguments whenever they need those arguments. These comments show that CBSS helped users in reducing interpersonal conflicts (isolated discussion, relieves intimidation, prevents personality interference) and supported them in resolving issue-based conflict (more time to think, reference to documented discussions).

Table 5.10 also shows examples of frustrated student comments. However, most of

them complained about the speed of time response or the time between sending a message and getting a response. They complained that CBSS is slow, slower than FTF, and they are unable to see facial expressions of their opposing group members while negotiating. The response time is slow because this time includes the time the opposing members read the message, the time to discuss the message with their fellows, and the time to type the response into the computer. The time that CBSS delivers the message from one side to the other is only a very small fraction of the users' time. From another point of view, CBSS gives the negotiators time to think well before responding, so that they focus their attention only on the problems, and try to give a reasonable response.

To reduce typing time, CBSS provides a way to read a note from a computer file previously prepared by the negotiators and to send this note to the other side. Negotiators may prepare notes using a wordprocessor program before conducting negotiations and save these notes as local files either on the computers' hard disk or on floppy disks. During the preparation stage, these notes can be transferred into a CBSS file system by opening the note using the wordprocessor and then performing a "copy" operation either for all lines or part of them. The note which resides in computer memory can be put into the note preparation window or into the comment editor window by performing "paste" operations, before saving or sending it. However, some students may not have fully used these features to save time. This was perhaps due to lack of familiarity with CBSS or because students did not have time to prepare notes before coming to the experiments.

It is true that having an isolated negotiation supported by CBSS removes body language communication, but this is the price that the users have to pay for a good separation

of people from problems, and to be able to conduct negotiation in distributed places. In addition, when it is needed, it is possible to use CBSS in parallel with Internet video conferencing software under a multi-tasking environment, which would cost only about $1,000 to install.

During the experiment some students experienced some technical difficulties, even though not all of these technical problems were caused by CBSS. Some of the problems related to CBSS were fixed during the experiment. Other students had mixed feelings about CBSS. They liked the fact that CBSS provided documentation and reduced personality issues, but they did not like the fact that it took a longer time to finish their bargaining tasks using CBSS than FTF.

To show how well CBSS documents the discussion, a clip from a discussion between a management team and a union team in reaching an agreement on the contract term issue is presented in Fig 5.6. As we can see, CBSS is able to record a negotiation discussion completely from the start to the end, and this documentation can be viewed or printed anytime. In traditional negotiation, even a very smart and fast secretary cannot provide the negotiators with complete documentation. A tape recorder might help with the task, but voice would need to be transcribed later to get hard copy. The transcription process may take longer, and unintentional errors may occur during the process.

Table 5.10 : CBSS user comments

Category	#Questionnaires	Example
No comment	16	
Satisfied	41	"I liked the fact that the information we discussed was at our finger tips if we needed to see it again. Moreover, it gave us a chance as a group to collectively discuss our responses, basically letting us think before we talked. Furthermore, the system went smoothly without any problems. On the whole I liked it much better than the face-to-face bargaining." "I liked the computer bargaining because it allowed our team to discuss issues and analyse their position without feeling intimidated by opposing members. Also we could see the entire communications at any time, which made it helpful for when we forgot some positions or items. Good system!!!"
Frustrated	14	"I would rather perform face-to-face bargaining than CBSS. CBSS is more frustrating and stressful. CBSS is too time consuming." "Frustratingly slow process, long wait for responses, couldn' t read reactions from bargaining partners. Unsure if arguments were received. Unclear response, couldn' t tell if other parties understood communication. Very difficult to move and use flexibility in complex arguments, because of slow response time (because info had to be typed). Other party didn' t have many arguments on disk, took a long time to think of responses and then type them in."
Technical	4	"The system works well. The only suggestion that I could make is that we often needed to write more than 60 characters for a proposal. If there is a way that the cursor could return to the "home" position it would reduce frustration and speed up the process. Looks good. Oh ya, if you could make it possible to send messages while the other team is using the common window, it would work much smoother (more user friendly)." "While using the "HotLine", when the cursor got to the end of the line I hit enter which sent the message rather than bringing the cursor down to the next line. This was annoying since in all the other windows hitting enter was required to bring the cursor down."

Mixed	9	"I think CBSS focus on the problem more if only tool used. I felt because I had FTF with other party I was already aware of personality issues. I believe that CBSS provides better documentation regarding analysis of negotiation strategies. I found that there was a disadvantage because you could not provide supporting document to help is swaying the opponent. Process slower because waiting for response also unaware if arguments received." "With using CBSS, we had time to discuss the issue with the group which was not possible in the face-to-face negotiations. It took longer using CBSS because of computer troubles, and in face-to-face it was much quicker."

[UNION-19:06] we want a one year deal
[MNGMNT-19:07] We laugh! We want a three year deal.
[UNION-19:07] what is your reason?
[MNGMNT-19:08] In 1995, 31% of all collective agreements were 36 months. Only 14.8% were 12 months. What makes you so special?
[UNION-19:11] we are a new union that is just starting out. we want to be part of the 15% because it is hard to say what the economy will be like in the future, next year
[MNGMNT-19:14] Your comments don't make sense. You guys are a new union just starting out. We don't know anything about you guys and we have no previous relationship with you that would allow us to give you a one-year deal. We need to establish a relationship with you in order to maintain stability.
[UNION-19:19] due to inflation we feel in our best interest we need a shorter contract so we benefit the most
[MNGMNT-19:21] Why should you benefit the most? Our concern is not for making the union money. We are willing to give you a 2 ½year contract. Please press ENTER before at the end of the window so we don't have to scroll you morons! Thank you
[UNION-19:25] 2 ½is unacceptable. we are willing to move to 1½like you want because we realize that we are young but you are new at this game also.
[MNGMNT-19:26] Let's cut to the chase. midpoint is 2 years. Will you agree?
[UNION-19:27] we want to finish this after we discuss wages. go there now
[MNGMNT-19:28] Forget it. You guys are being ridiculous. Let's finish this. Two years or this may get nasty.
[UNION-19:30] we will accept 1½years based on economy and the inflation, this will benefit you and I
[MNGMNT-19:31] We didn't offer 1½ Two ... years - remember this is ONLY you first contract and the majority of contracts are 2 years
[UNION-19:32] what other reasons do you have for wanting two years? be specific?
[MNGMNT-19:36] Speaking of specific - what the hell are you talking about when you refer to the economy? We gave you stats - 31% of contracts are 3 years in the public sector, 14.8 are 1 year. 2 years is 32.9%. In addition, we don't want an agreement of 1 year because we have no knowledge of your previous bargaining record. We want to gain a sense of trust first and maybe in two years we'll talk
[UNION-19:41] the economy now is weak and we do not want to be locked into long term deals. if the economy goes down, you are still paying constant wages.
[MNGMNT-19:43] How long is 2 years? If the economy goes down, we won't be able to afford to pay you any wages. Let's just agree and move on.
[UNION-19:44] we will agree, contingent upon the wage negotiation. be quiet
[UNION-19:46] we realize wages are not this issue, but the total contract deals with issues other than just contract length. be flexible. we are willing to accept a 2 year deal, but keep in the back of your mind that when it comes to wages, you can't be this stingy. we can hear you, so keep it down
[MNGMNT-19:45] Wages are a separate issue. Is two years okay?
[UNION-19:49] hurry UP
[MNGMNT-19:49] Fine, 2 years is affirmed.
[UNION-19:49] remember wages
[UNION-19:49] go to wages window

Figure 5.6 An example of negotiation on the contract terms

5.4.5 Analysis of the Second Experiment Results.

Questionnaire Q3 was distributed to respondents after conducting the second experiment. Respondents were asked to give comments or opinions to seven questions included in the questionnaire. The responses are summarized in the following paragraphs.

Question #1 : As a system to support the negotiation process, does CBSS offer advantages over face-to-face (FTF) negotiations?

Response summary: Respondents agreed that CBSS "has some advantages over FTF negotiations, especially if the parties are geographically separated and if the parties are too emotionally involved". We can imagine situations where FTF negotiation is not possible because the members of parties are very busy with their business around the world, or they are moving from one place to other places continuously. One way they can negotiate in such an environment is through a Web-based NSS such as CBSS. Other advantage are "automatic documentation and private discussions". Automatic documentation is very helpful when the negotiators need reference to previous discussions, especially when the negotiation is conducted on non- consecutive days. Chances to discuss privately among members of a party without being monitored by other parties, and without leaving the negotiation sessions is another advantage of using CBSS. Since CBSS separates people from the problems, the negotiators can focus more on issues rather than on personalities (less anger involved, less intimidation). CBSS will help some people who are shy and express themselves better in writing than in speaking.

Question #2: What are the disadvantages of using a system like CBSS in conducting negotiations?

Response summary: Most respondents thought that "by using CBSS the negotiators lose non-verbal language communication, they were frustrated in waiting for responses, there were delays in hearing from the other negotiators, and there is a technology dependence on using CBSS". Non-verbal language may enrich negotiator communication during face-to-face negotiations, but mis-interpretation of non-verbal language may also happen, which may lead negotiators to wrong conclusions. If negotiators really need to see each other during the sessions then CBSS can be run simultaneously with video conferencing software such as Cu-SeMee or PreVue. Delays and slow responses from other negotiators depend on the human abilities of CBSS users such as the ability to read fast, to type fast, and to interpret comments fast. CBSS does not delay messages exchanged during the discussion. Technology dependence is not an important issue, because if we like to use the technology then we will depend on it.

Question #3 : Do you think that CBSS can be used as an alternative to conducting negotiations if face-to-face meetings are not possible?

Response summary: All respondents were positive, with comments such as: "it is better to use CBSS rather than waiting until face-to-face meetings are possible; CBSS is suitable for international company negotiations; when parties are not in close proximity then CBSS is a good alternative; most negotiators probably prefer to meet face-to-face whenever possible".

Question #4: Do you think it is possible to use CBSS in conjunction with face-to-face negotiation?

Response summary: The respondents answered yes, with various comments. Some respondents said that "CBSS can be used when there is a need to continue negotiations that have become too personal, it is better to try face-to-face first, if it does not work then CBSS might be worth a try". Some others said that "CBSS is good for documenting the discussions, and it is possible to use CBSS as a complement to face-to-face negotiations". The others said that "CBSS is a support tool, and not a substitute, but it is good for the early stage of negotiations".

Question #5 : Would you prefer to use CBSS rather than face-to-face?

Response summary : There were mixed responses, with some saying yes and others saying no. One respondent said "no without further explanation", another said "no because non-verbal feedback is important in the process". The rest of the respondents said yes, with comments such as: "I prefer CBSS if face-to-face would reveal attitude and personality conflicts occurring; I would prefer CBSS if I was going up against experienced negotiators; and it depends on subject of negotiations and the relation with other parties".

Question #6 : Do you think that it is possible to get better outcomes using CBSS rather than face-to-face negotiations?

Response summary: It is interesting to note that most respondents thought the outcomes do

not depend on the method used, because this finding supports the results of hypothesis H4 and H5 tests. Some of the responses were: "the results would be about the same; I think the outcomes will be more or less equal; you would get similar results".

Question #7 : Collective bargaining is often a time consuming process. Would the use of CBSS as a support system increase or decrease the length of the bargaining process?

Response summary: Some respondents said that "CBSS would increase the negotiation time because a longer time is needed to type and to read messages, and because CBSS allows negotiators to take as much time as they want in replying to messages". It is interesting that other respondents said that "CBSS would decrease the bargaining time because the negotiators do not need to leave the bargaining table to have caucus meetings, or because CBSS offers the parties the flexibility to negotiate, away from formal meetings".

It is interesting to see that the second experiment also revealed a quite positive response to CBSS. In general, the respondent' s opinions confirmed and provided insights to explain the test results of the first experiment.

5.5 CBSS Usability.

The system design and implementation of CBSS was based on the system requirements (including usability requirements) which were presented in section 4.2 (System Specifications). All requirements specified in that section was satisfied in the CBSS

implementation. CBSS is a windows oriented program implemented by using the Java programming language GUI standard. This GUI standard was also used to implement the WISIWYS requirement. The system was tested by using the existing computer network at McMaster University. Respondents agreed that CBSS was user-friendly computer software (Q1-12).

Documentation support was implemented by having a reliable data and file management module in the CBSS implementation. The CBSS server system handles the data and file management. All discussions during the bargaining process are saved in appropriate files, which are backed up periodically by the file management module. When the server system receives data or messages from the clients, these data or messages will be appended immediately into their corresponding files. In case of system failures, only the most recent message, which has not been saved by the user, will be lost. The file management provides an automatic easy recovery process if errors do occur.

CBSS provides an easy-to-use menu, which is a point-and-click oriented menu. The CBSS client system is executable under a Java enabled Web browser running on any multi-tasking computer platform. All the usability requirements were implemented in order to have a system which is easy-to-use and to learn with minimal errors.

The experiments were completed successfully, since the negotiation teams signed final agreements at the end of their simulated collective bargaining processes under CBSS. Users of CBSS were students who had never tried to use or observed CBSS before the experiments. These users were given only minimal training (30 minutes) and allowed to try the system for

30 minutes, before beginning negotiation. This is a good sign that CBSS is very easy to use and to learn. Users agreed that CBSS is user-friendly software (Q1-12), which is another proof that CBSS is easy to use and to learn. (Note: the results of the statistical analysis for questions in questionnaire Q1 are shown in Table 5.9, and for questions in questionnaire Q2 in Table 5.7A). The implementation of CBSS system requirements is summarized in Table 5.11.

The final agreements that student team produced at the end of time-limited experiments also show that to some degree CBSS was efficient. Once the users learned how to use CBSS, they were able to use the system to produce final agreements. Another proof that CBSS is an effective tool was that the users agreed that CBSS helped them in preparing final agreement (Q1-5), and CBSS made valuable contributions to negotiation outcomes (Q1-6).

Unfortunately, there was no chance to observe the memorability property of CBSS, because the users had only one chance to use the system. Probably in further research this property will be observed.

In terms of mishandling of data, CBSS had a low error rate. The events that in some cases frustrated the users were occasional broken communication channels between the two parties. Since CBSS relies on the availability of Internet links, sometimes the CBSS communication channel failed when Internet was too busy. The system did not crash but the communication channel needed to be re-established. During the recovery process, it was observed that this was done quickly and there were no data losses.

User satisfaction can be inferred from the conclusion that users would not avoid using CBSS (Q1-14), they would use CBSS in conjunction with FTF (Q2-19), and they would use CBSS if FTF was not possible (Q2-20).

Table 5.11 System requirements and their implementation.

System Requirements:	Implementation:
the system must implement a Graphical User Interface (GUI) standard	-multi windows system -mouse activated system -Java GUI standard
the system must provide the "What I See Is What You See" (WISIWYS) interfaces on public windows	-ideas and messages are distributed by server system to clients without modifications
the system must be able to provide parallel two-way communication between the parties	-communication is supported by TCP/IP on the Internet -server system manages the communication among clients
the system must provide coordination in ideas exchanges and message exchanges	-server system coordinates the activities
the system must have reliable data and file management support	-clients and server cooperate in data and file management
the system must prevent most errors from occurring, and must provide easy recovery if errors do occur.	-server system provides backup automatically
the system must provide user menu.	-Pre-Session, Session, and Help menu
the system must be specially designed for the negotiation process	-structures the process into Pre-Session and Session
the system must be able to utilize existing computer networks	-it uses existing computer network connected to the Internet
the system must be executable under a Web browser	-the client system is executable under a Web browser
the system must be implemented as a Client/Server system	-CBSS consists of a server system and a client system
the system must be an architecturally neutral system.	-Java and the Web guarantee platform independency

Chapter 6. Conclusions and Further Development.

The prototype of CBSS, a Web-based negotiation support system was developed, implemented, and tested. The CBSS program was written in the Java language, designed as a Client/Server system, and adopted the World Wide Web as its networking platform. Two experiments were conducted to evaluate the validity of CBSS as an alternative to face-to-face negotiations, and to investigate the effectiveness and efficiency of CBSS as a negotiation support system. The results of the experiments show that negotiating with CBSS is a valid conditional alternative to face-to-face negotiation. Although bargaining processes supported by CBSS are perceived to be slower than face-to-face negotiation, CBSS is an effective tool for negotiators, and the end results of using CBSS are as good as face-to-face negotiation.

CBSS is computer software designed to support the collective bargaining process, either in combination with face-to-face meetings, or in a distributed-synchronous meeting. CBSS does not provide suggestions or solutions to the users but provides process support, especially through remote communication, documentation handling, and process structuring.

CBSS was designed as an adversarial group meeting support system, and differs from existing EMS-based support to negotiation. CBSS utilizes an existing computer network such as a LAN attached to the Internet and it can be accessed through the Web in a teleconferencing setting. In this way CBSS differs from the existing process support NSS which require sophisticated meeting room settings.

Some advantages of using CBSS over the traditional bargaining process are: parallel communication, automatic documentation, structured discussion, and possibly process and cost optimization. Communications through computer terminals lack a non-verbal communication channel, but it tends to be less expressive, more de-personalized and businesslike, and it tends to help in separating people from the problems. CBSS can be used as an enhancement to face-to-face negotiation and as a supporting tool in distributed-synchronous negotiation. CBSS structures the bargaining process, making the process easy to organize, and the software is user-friendly. CBSS can also be used when FTF meetings cannot be arranged, e.g., when bargaining teams are required to travel long distances. Also, it allows geographically dispersed teams to caucus when meeting is impractical.

The current implementation of CBSS is a text-based support for negotiation, which means that negotiators communicate with each other through written messages. Most people are more careful in writing than in speaking. For example, during the experiments it was observed that CBSS users performed proof-reading, made corrections, and made sure that the message were clear, understandable, and relevant to the issue discussion, before sending the message to the other party. The other side negotiators always read the message carefully before making a reply. In using CBSS as a negotiation support system, negotiators are allowed to think before replying arguments. In this case, CBSS actually improve the quality of discussions and made the discussion more thoughtful than in FTF negotiation.

In addition, CBSS provides structured documentation so that negotiators can locate discussion files easily. In this case, CBSS allows negotiators to do more extensive study and to analyze their bargaining concessions or their bargaining strategies. In face-to-face negotiation, the words are spoken only once, and there is no way to go back to previous discussions, precisely as CBSS provided, for doing analysis and extensive study.

This research presents the first study that compares face-to-face negotiation with isolated CBSS negotiation. Previous studies only presented the comparison between face-to-

face negotiation without computer support and face-to-face negotiation with computer support. In addition, CBSS is the first support-type NSS deployed on the Web with a teleconference setting. Existing support-type NSS were implemented on a specific LAN with a decision room setting. In this case, CBSS implementation is cheaper and more accessible than the existing support-type NSS.

As we know, most Web contents are passive and static information, and the implementation of CBSS proves that a system which combines interactivity with dynamically changing Web pages can be implemented. Furthermore, CBSS is easier to implement, because it requires no installations on the client machine, which means that it eliminates the problems of software upgrading and maintenance.

In the current implementation of CBSS there are no personal DSS modules. However a separate DSS module can be used in parallel with CBSS under a Windows '95 environment. Further development can include a security module, audio and video components, and a fuzzy model for negotiation. The security module can be used to protect documentation and the negotiation process from unauthorized access. The audio and video components can enrich communication among the parties. A fuzzy model can present solution alternatives to the users, based on fuzzy data concerning the issue attributes. Further studies on the positive and negative impacts of these potential functions are necessary. A third party mediation function has been built into the system but has not been tested yet. This also needs further experimentation to measure its usefulness.

Besides being a system to support the negotiation process, CBSS can also be used as an educational tool, especially for laboratory experiments on the negotiation process. The very detailed documentation provides the opportunity for an instructor to do sophisticated analysis on the negotiation strategies used by negotiators.

References:

Adie, C., Distributed multimedia information systems, *Computer Networks and ISDN Systems* 25 (Suppl.2) (1993) S49-S57.

Anson,R.G. and Jelassi,M.T. (1990). "A development framework for computer-supported conflict resolution", *European Journal of Op.Research* 46, 181-199.

Applebaum, M.I., and Cramer, E.M. (1974). "Some problems in nonorthogonal analysis of variance", Psychological Bulletin 81(6) 335-343.

Burke,K. and Chidambaram,L. (1995). "Developmental differences between distributed and face-to-face groups in electronically supported meeting environments: An exploratory investigation", *Group Decision and Negotiation*, 4, 213-233.

Carmel,E and Herniter,B.C. (1989). "MEDIANSS: Conceptual design of a system for negotiation sessions", *Proceeding 9th Conf. on DSS*, DSS-89 Transaction, 239-253.

Carmel, E., Herniter, B. C., and Nunamaker Jr, J.F. .(1993). "Labor-management contract negotiation in an electronic meeting room: A case study", *Group Decision and Negotiation*, 2, 27-60.

Chorafas, D.N., *Beyond LAN: Client/Server Computing*, McGraw-Hill Series on Computer Communications, New York: McGraw-Hill, Inc., 1994.

Crawford,V.P. (1985). "Dynamic games and dynamic contract theory", *Journal of Conflict Resolution* 29:2, 195-224.

Cronbach, L.J. (1951). "Coefficient alpha and the internal structure of tests", Psychometrika, 16, 297-334.

Daniel, Dianne., "Watch for a new king in the next frontier", (1996), http://www.plesman.com/cc/csc/pg3.htm.

Dennis,A.R, George,J.F., Jessup,L.M., Nunamaker Jr,J.F. and Vogel,D.R. (1988). "Information technology to support electronic meetings", *MIS Quarterly*, December, 591-615.

DeSanctis,G. and Gallupe,B. (1989). "Group Decision Support Systems: A new frontier", In Sprague(Jr.) ,R.H and Watson,H.J.(eds), "*Decision Support System: Putting theory into practice*", Prentice Hall, Englewood Cliffs, New Jersey.

Deutsch,M., (1969)."Conflicts: productive and destructive", In Jandt,F.E.(ed.), "*Conflict resolution through communication*". New York:Harper and Row.

Dewire, D.T., *Client/Server Computing*, New York, N.Y.: McGraw-Hill Inc., 1993.

Executive Decision Services, *Policy PC Version 2.0 Reference Manual.*

Executive Software, Inc., (1983). *Decision Analyst User Manual.*

Fisher,R. and Ury, W. (1981). *Getting to Yes: Negotiating agreement without giving in*, Penguin, New York, NY.

Foroughi, A., Perkins, W.C., and Jelassi, M.T., (1995). "An empirical study of an interactive, session-oriented computerized Negotiation Support System (NSS)", *Group Decision and Negotiation*, 4: 485-512.

Fraser, M.N. and Hipel, K.W., (1981). "Computer assistance in labor-management negotiations", *Interfaces* 11, 22-29.

Fraser, M.N. and Hipel, K.W., (1984). *Conflict analysis: models and resolutions*, North Holland Series in System Science and Engineering, Elsevier Science Pub.Co.Inc, 159-174.

Fraser, M.N., (1993), *The power of game: Managing the fundamental forces controlling human relationships*, Dept.of Managment Science Univ.of Waterloo, Nov.1993.

Gershenfeld, J.C., McKersie, R.B., and Walton, R.E., (1995), *Pathways to change: Case studies of strategic negotiations*, W.E. UPJOHN Institute For Employment Research, Kalamazoo Michigan.

Gray,P. and Nunameker,J.F. (1989). "Group Decision Support System", In Sprague(Jr.) ,R.H and Watson,H.J.(eds), *Decision Support System: Putting Theory into Practice*, Prentice Hall, Englewood Cliffs, New Jersey.

Greer, D.J., "Client/Server, the Internet, and WWW", Robelle Consulting Ltd, 1995, http://www.netstorage.com/wwwpaper.html.

Gulliver, P. H. (1979). *Dispute and negotiation: A Cross-cultural perspective*. Academic Press, NewYork.

Jarke,M and Jelassi,M.T. (1986). "View integration in Negotiation Support System". In Fedorowics,J [ed.], *DSS-86 Transaction*, 6th Int. Conf. on DSS, Washington April 1986.

Jarke,M., Jelassi,M.T. and Shakun,M.F. (1987). "MEDIATOR: Toward a Negotiation Support System", *European Journal of Op.Research*. 31, 314-334.

Jelassi,M.T. and Foroughi,A. (1989). "Negotiation Support System: An overview of design issues and existing software", *Decision Support System*, 5, 167-181.

Kessler, Sheila. (1978). *Creative conflict resolution: Mediation leader's guide*, National Institute for Professional Training, Fountain Valley, CA.

Kersten,G.E. (1985) "NEGO - Group Decision Support System", *Information & Management*, 8, 237-246.

Kersten, G., Matwin, S., Michalowski, W., and Szpakowicz, S. (1986). "Rule-based modelling of negotiation strategies", *Working Paper Series*, Carleton University.

Kochan,T.A. and Verma,A. (1983). "Negotiations in organizations:blending industrial relations and organizational behavior approaches", In Bazerman,M.H and Lewicki,R.J (eds), *Negotiation in Organization*, Sage Publications, Beverly Hills.

Lee, T.Berners et.al., "The World Wide Web", *Communications of the ACM*, August 1994/Vol.37, No.8, 76-82.

Lim,Lai-Huat and Izak Benbasat. (1993). "A Theoretical perspective of Negotiation Support System", *Journal of MIS*, Winter 1992-93, 9(3), 27-44.

McCartt, A.T. and Rohrbaugh,J. (1989). "Evaluating Group Decision Support System effectiveness: A performance study of decision conferencing", *Decision Support Systems* 5, 243-253.

McGrath, J. (1984). Groups, Interaction and Performance. Englewood Cliffs, NJ: Prentice-Hall.

Miranda,S.M. and Bostrom,R.P. (1994). "The impact of group support systems on group conflict and conflict management", *Journal of MIS*, Vol.10, 63-95.

Mumpower, J., Schuman, S., and Zumbolo, A., (1986). "Analytical mediation: An application in collective bargaining", *Working Paper Series*, State University of New York at Albany.

Nielsen, Jacob. (1993). *Usability Engineering*. Academic Press, Inc. Harcourt Brace & Company, Publishers, Boston.

Norris,C.N. and Rolph, J.E., (1981). *Introduction to Data Analysis and Statistical Inference.* Prentice-Hall Inc. Englewood Cliffs, NJ.

Nunamaker,J.F. et.al. (1991). "Electronic meeting systems to support group work", *Communications of the ACM*, Vol.34, No.7, 40-61.

Paolo, C. et.al., "An architecture to coordinate distributed applications on the Web", *Computer Networks and ISDN Systems*, 28 (1996) p.941-952.

Phillips, G.E., (1977), *The Practice of labour relations and collective bargaining in Canada*, Butterworth and Co. Ltd, Toronto.

Pickett, John (editor), Focus Issue: Client/Server World, June 1996, page 5.

Pratt, M.M., "Java and the Client/Server environment", (1996), http://www.unex.ucr.edu/gis/prat.htm.

Pruitt,D.G. (1993). "Achieving integrative agreements", In Bazerman,M.H and Lewicki,R.J (eds), *Negotiation in Organization*, Sage Publications, Beverly Hills.

Rahim,A. (1985). "A Strategy of managing conflict in complex organization". *Human Relations*, 38, 1 , 81-89.

Raiffa,Howard. (1982). *The Art and Science of Negotiation*, The Belknap Press of Harvard Univ.Press, Cambridge.

Rojot,Jacques. (1991). *Negotiation: from theory to practice*, Macmillan Academic and Professional Ltd, Houndmills.

Schussel, G., "Client/Server, from dBASE to Java is it over, or just the beginning?", (1996), http://www.dciexpo.com/geos/java.htm.

Shapiro, S.S., and M.B. Wilk, (1965), "An analysis of variance test for normality (complete samples)", *Biometrika*, vol. 52, pp. 591-611.

Short,J., E.Williams, and B.Christie. (1976). *The Social Psychology of Telecommunications*, London: John Wiley & Son.

Sprent,P., (1989). *Applied Nonparametric Statistical Methods.* Chapman and Hall, London.

Walton,R.E. and McKersie, R.B., (1965). *A Behavioral Theory of Negotiations*, McGraw-Hill Book Company, New York.

Watson, R.T., DeSanctis, G., and Poole, M.S. (1988). "Using a GDSS to facilitate group concensus: Some Intended and Unintended Consequences", *MIS Quarterly*, Sep. 1988, 463-477.

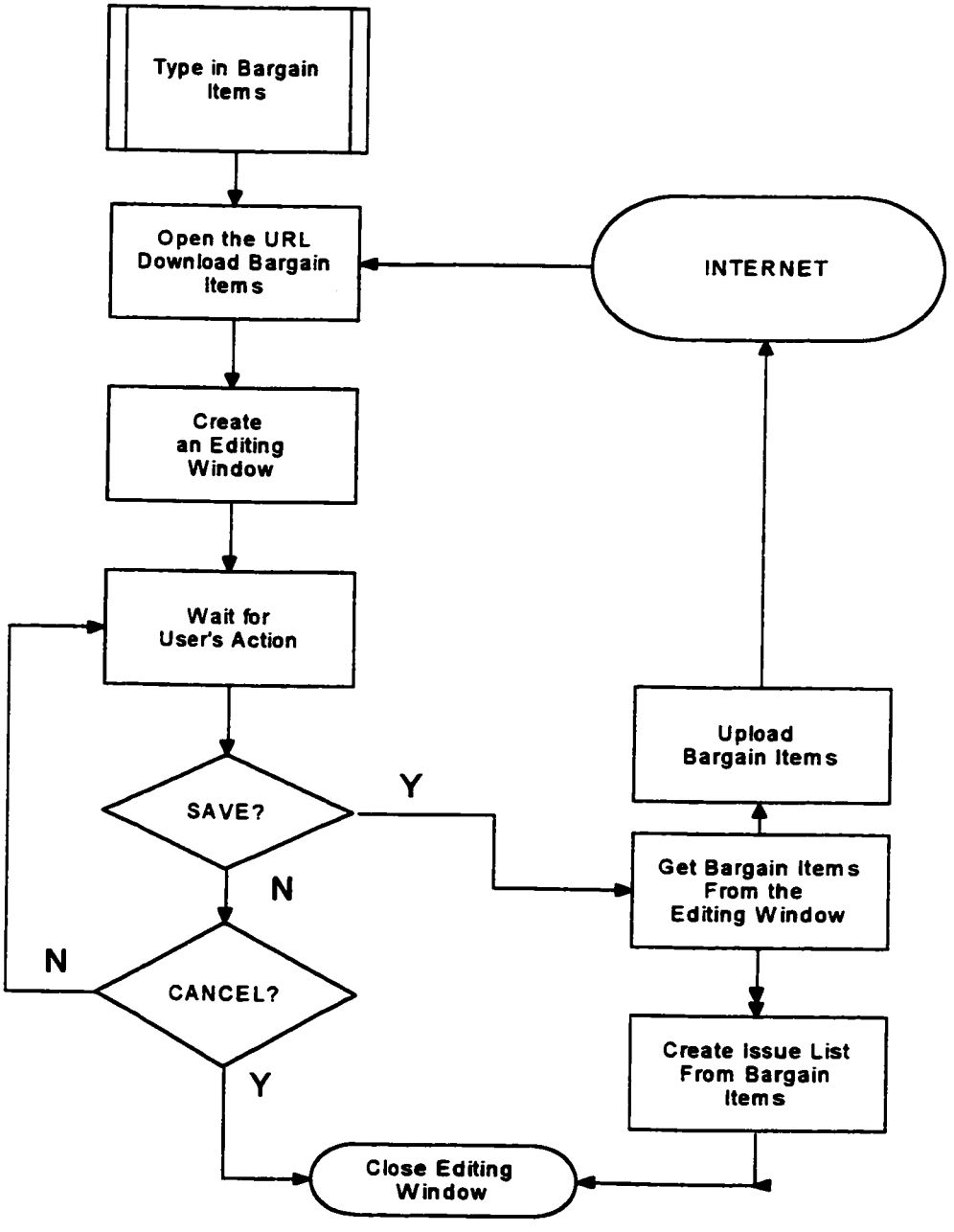
Weiers, R.M., (1991). *Introduction to Business Statistics*. The Dryden Press, Chicago.

Winter, F., (1985). "An Application of computerized decision tree models in management-union bargaining", *Interfaces* 15, 74-80.

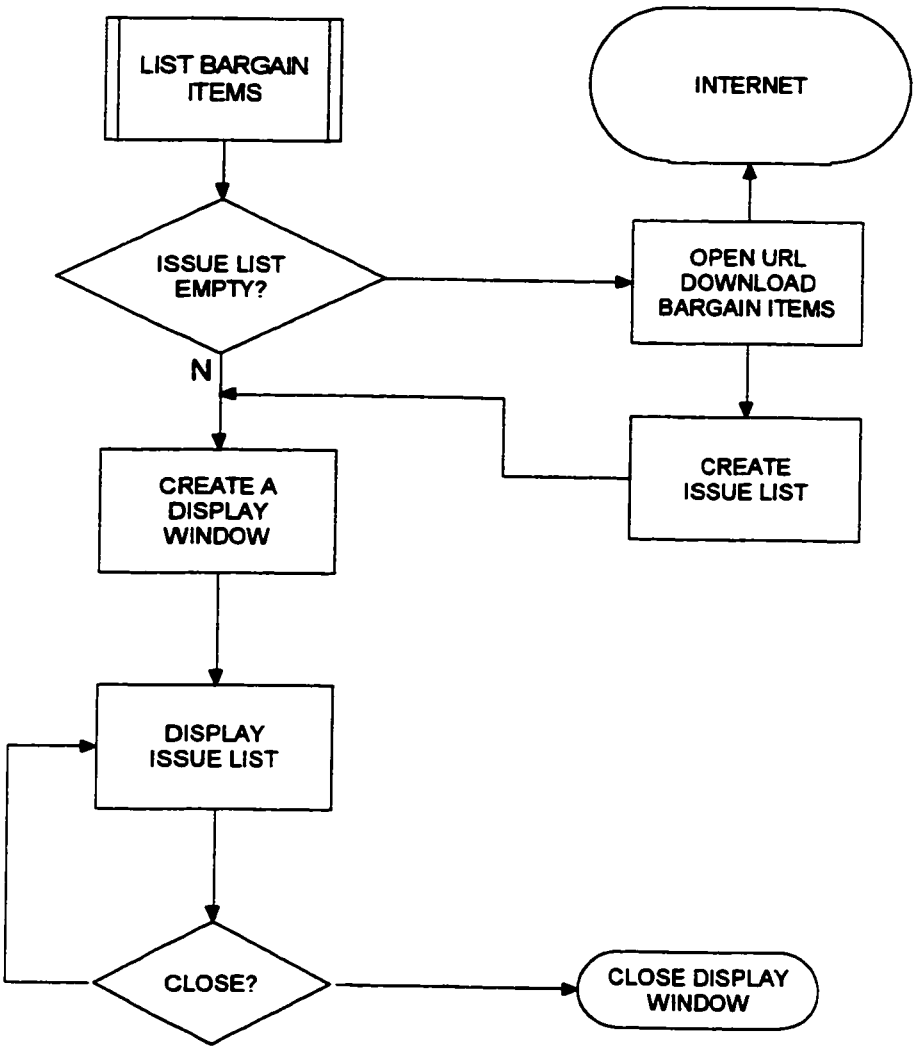
Zeidenberg, Jerry., "Client-Server fuelling the IT operation", (1996), http://www.plesman.com/cc/csc/pg7.htm.

Zeleny, Milan., (1982). *Multiple Criteria Decision Making*, McGraw-Hill Book Company, New York, 99-127.

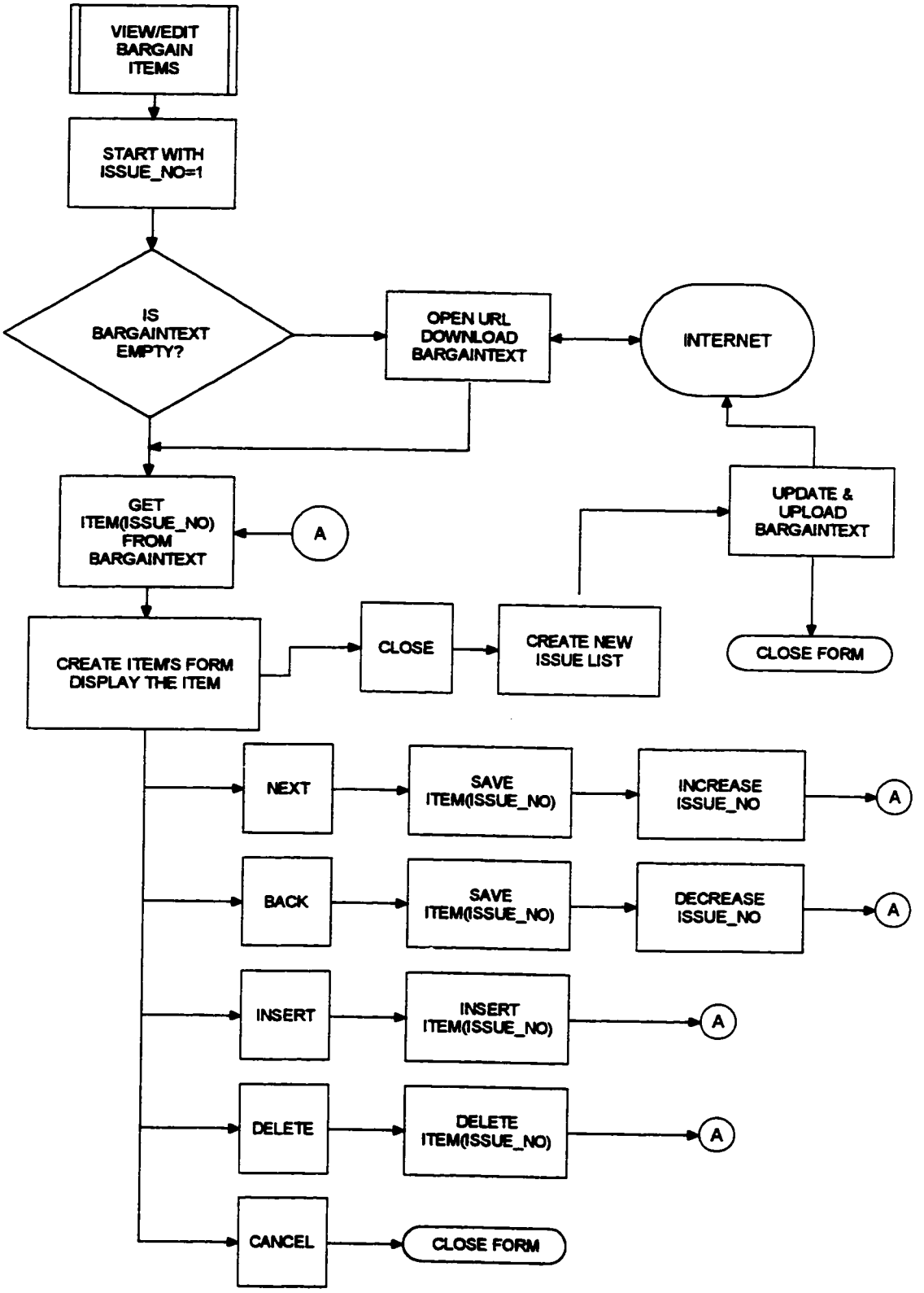
Appendix A. System Flowchart



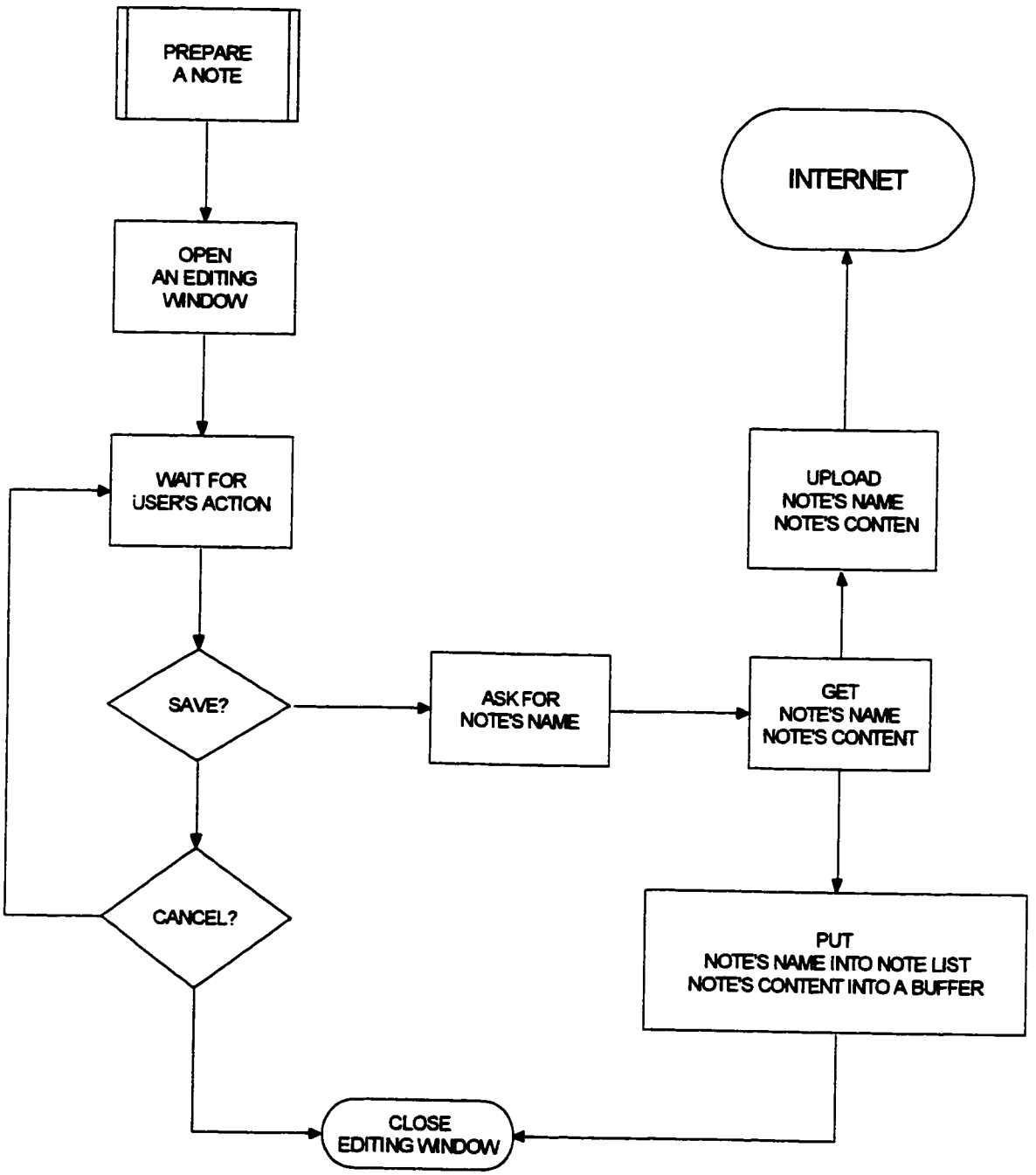
A1. Type-in Bargain Items Flowchart



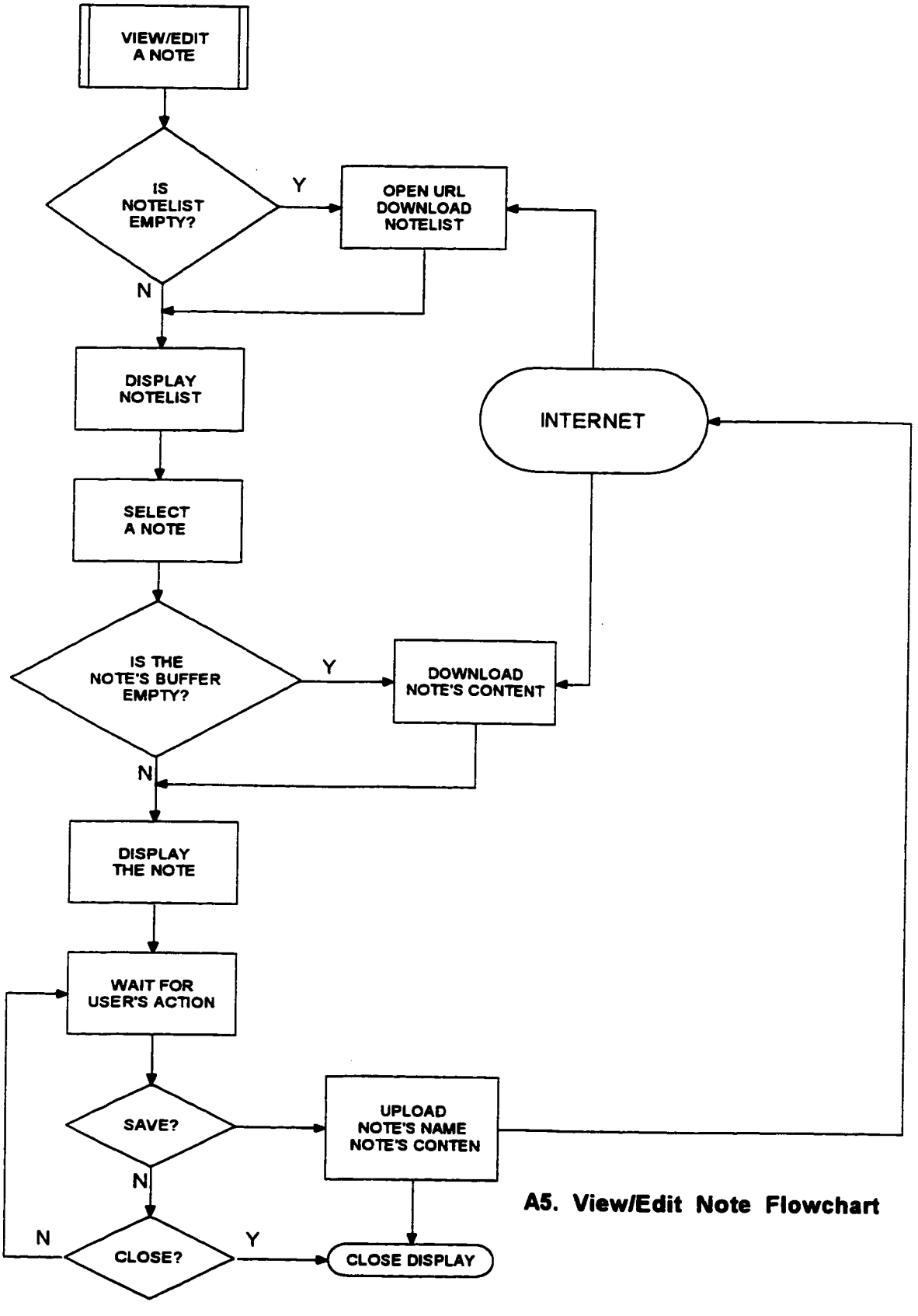
A2. List Bargain Items Flowchart



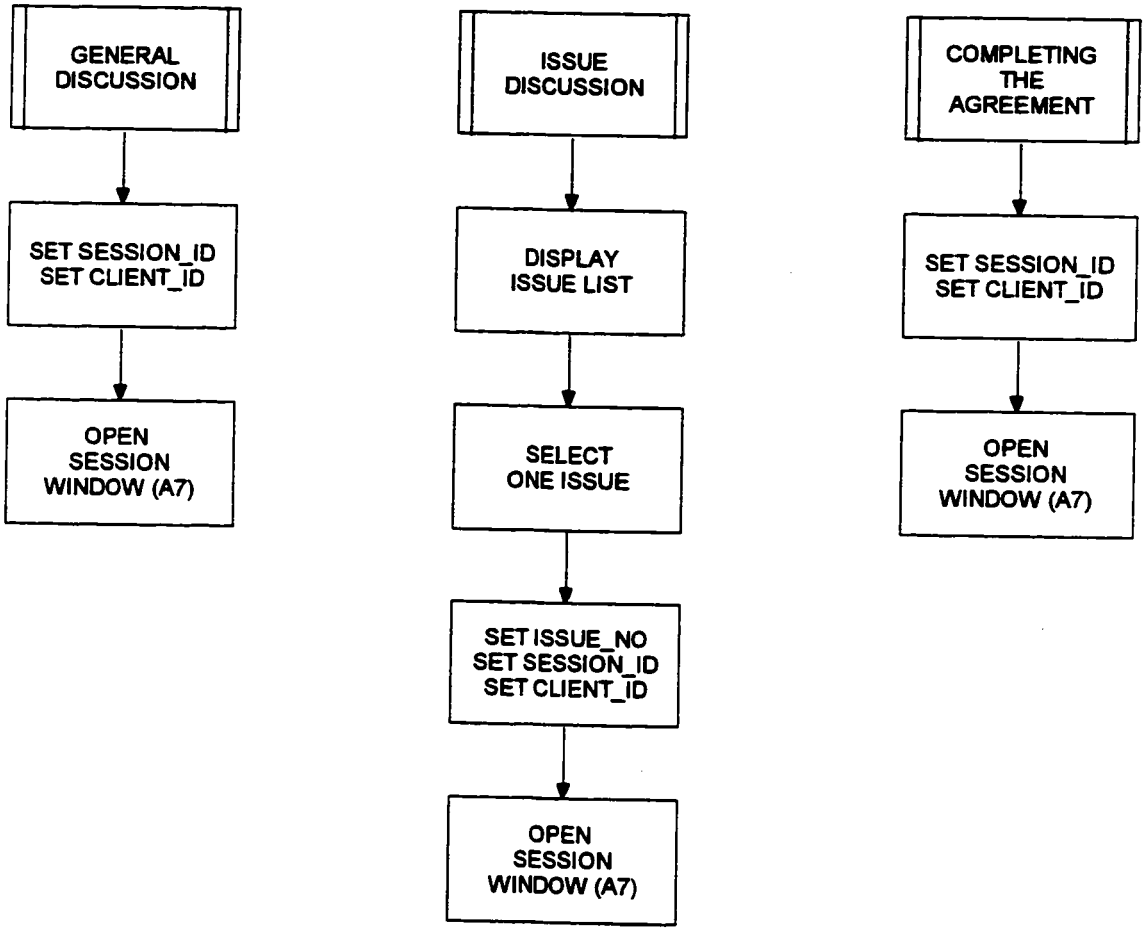
A3. View/Edit Bargain Items Flowchart



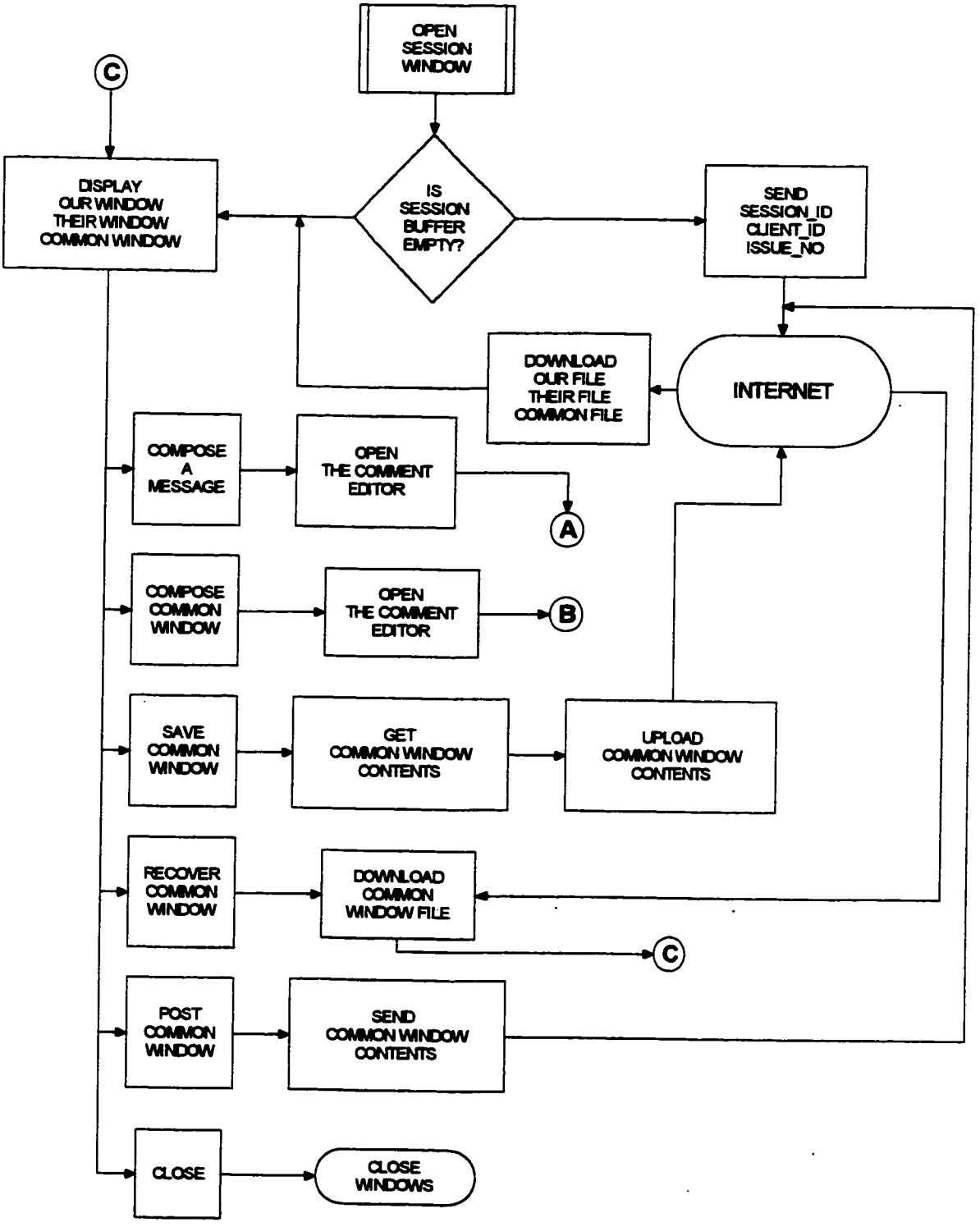
A4. Prepare a Note Flowchart



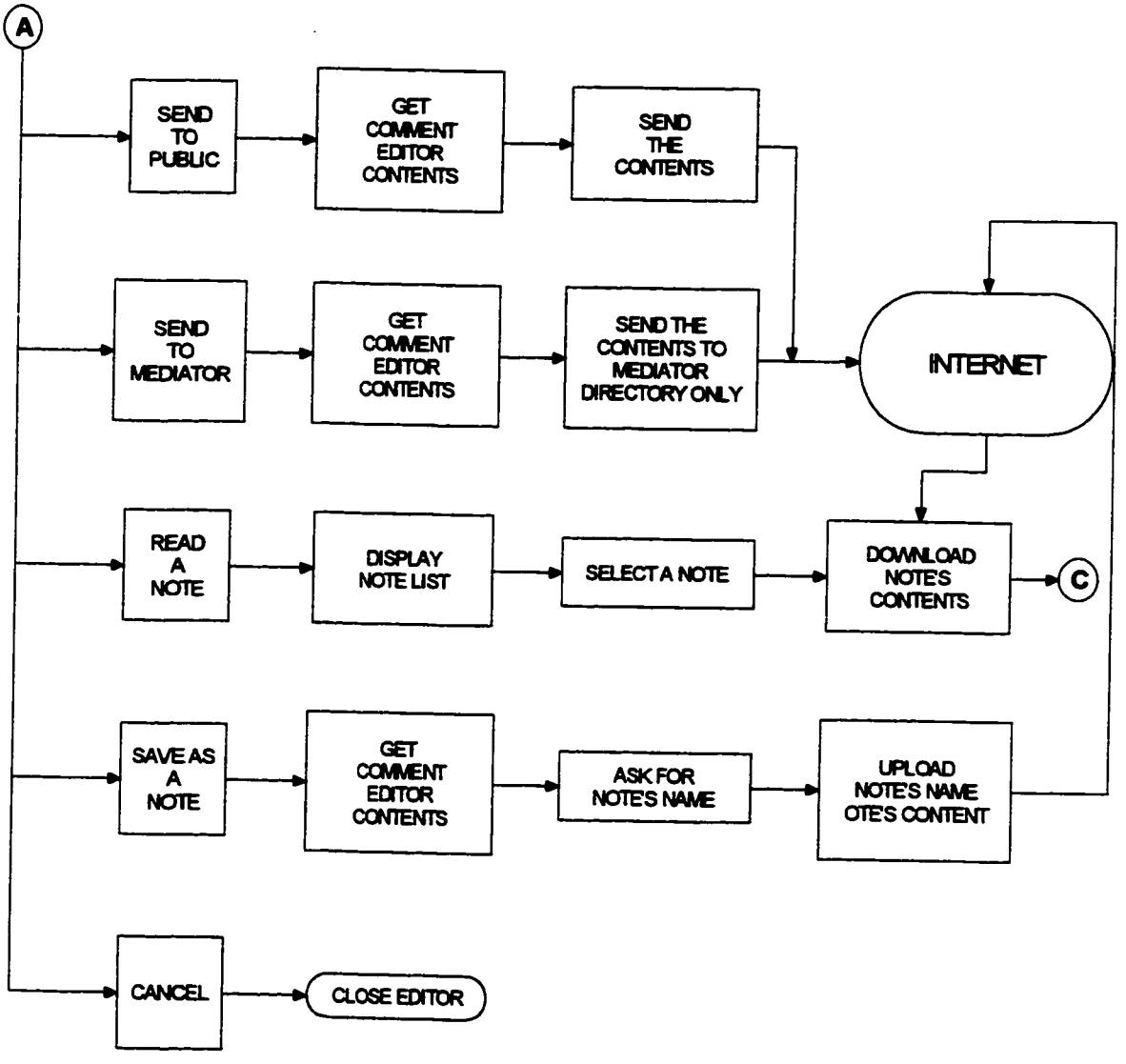
A5. View/Edit Note Flowchart



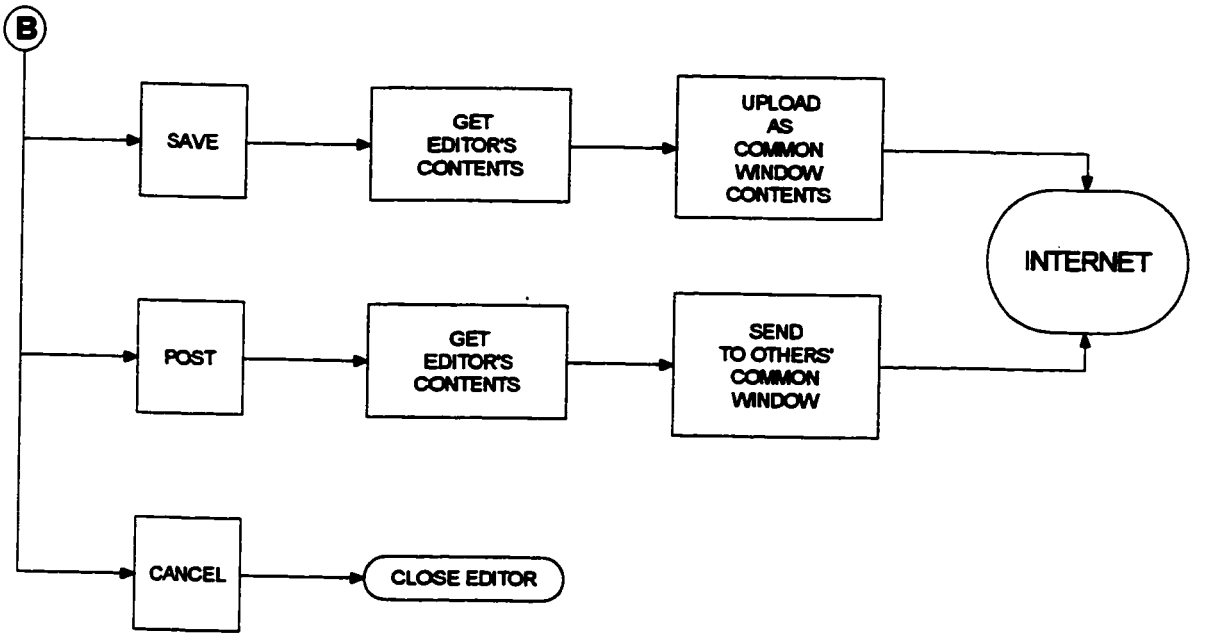
A6. Sessions Flowchart



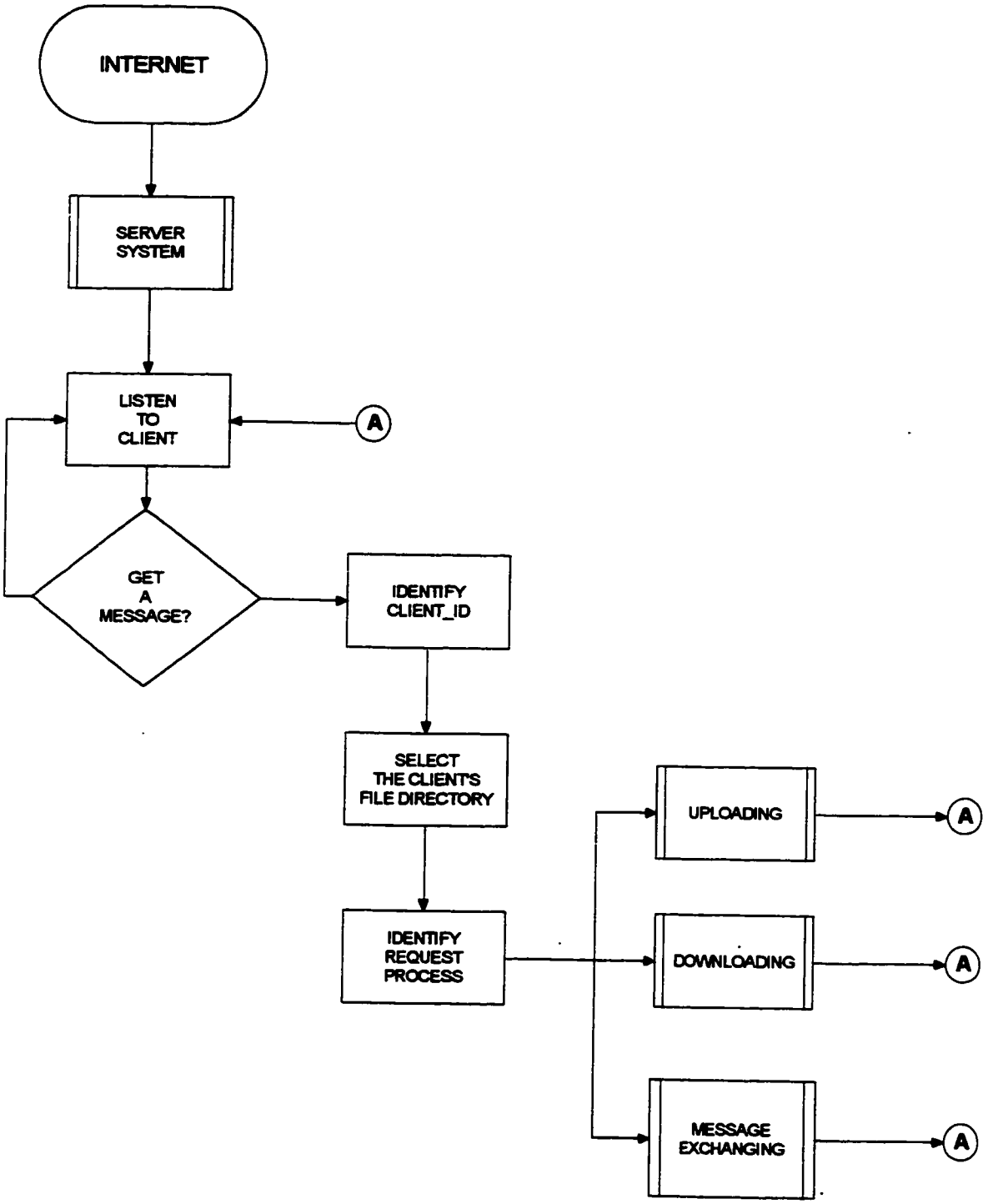
A7. Open Session Windows Flowchart



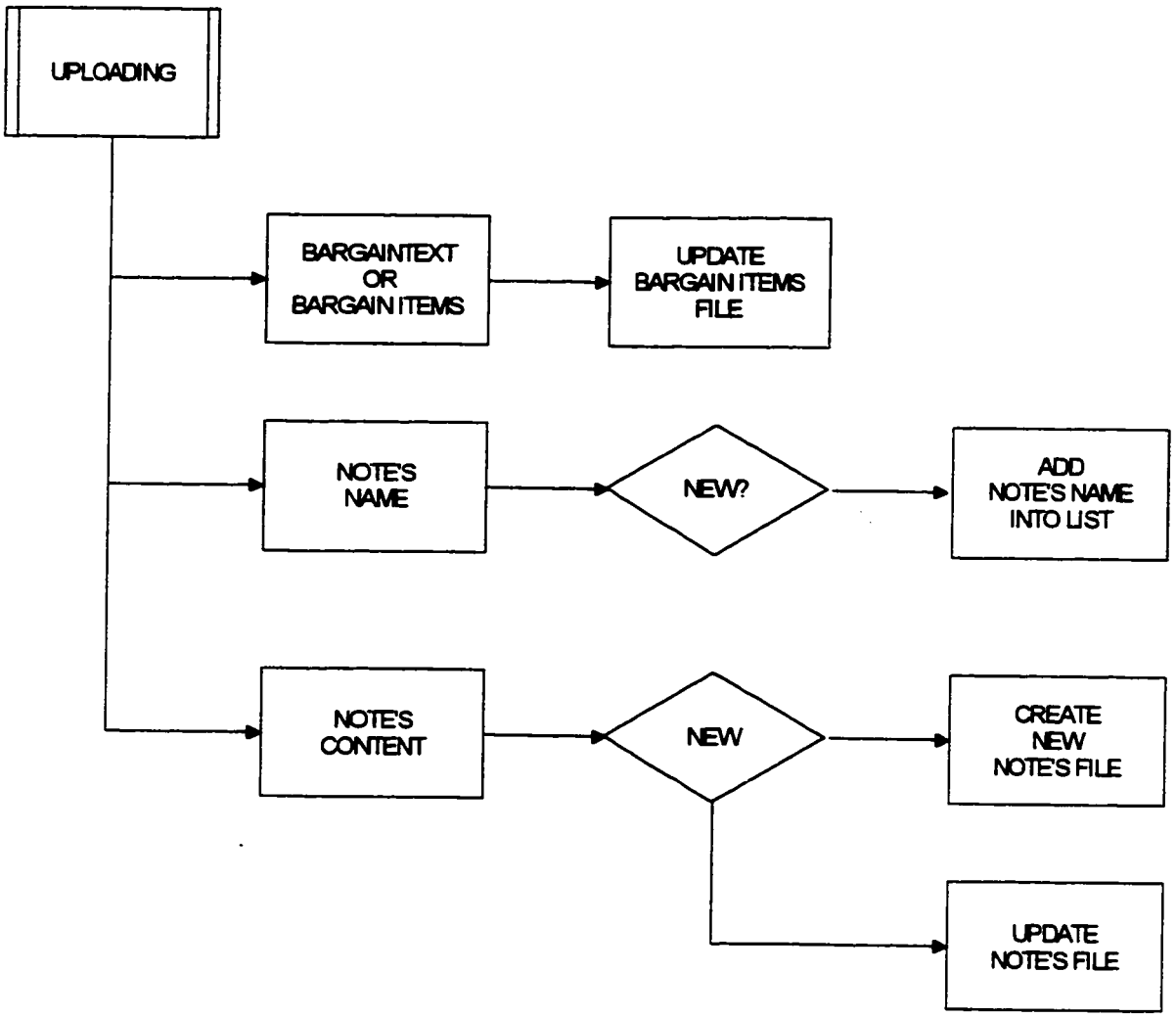
A7. Open Session Windows Flowchart (cont)



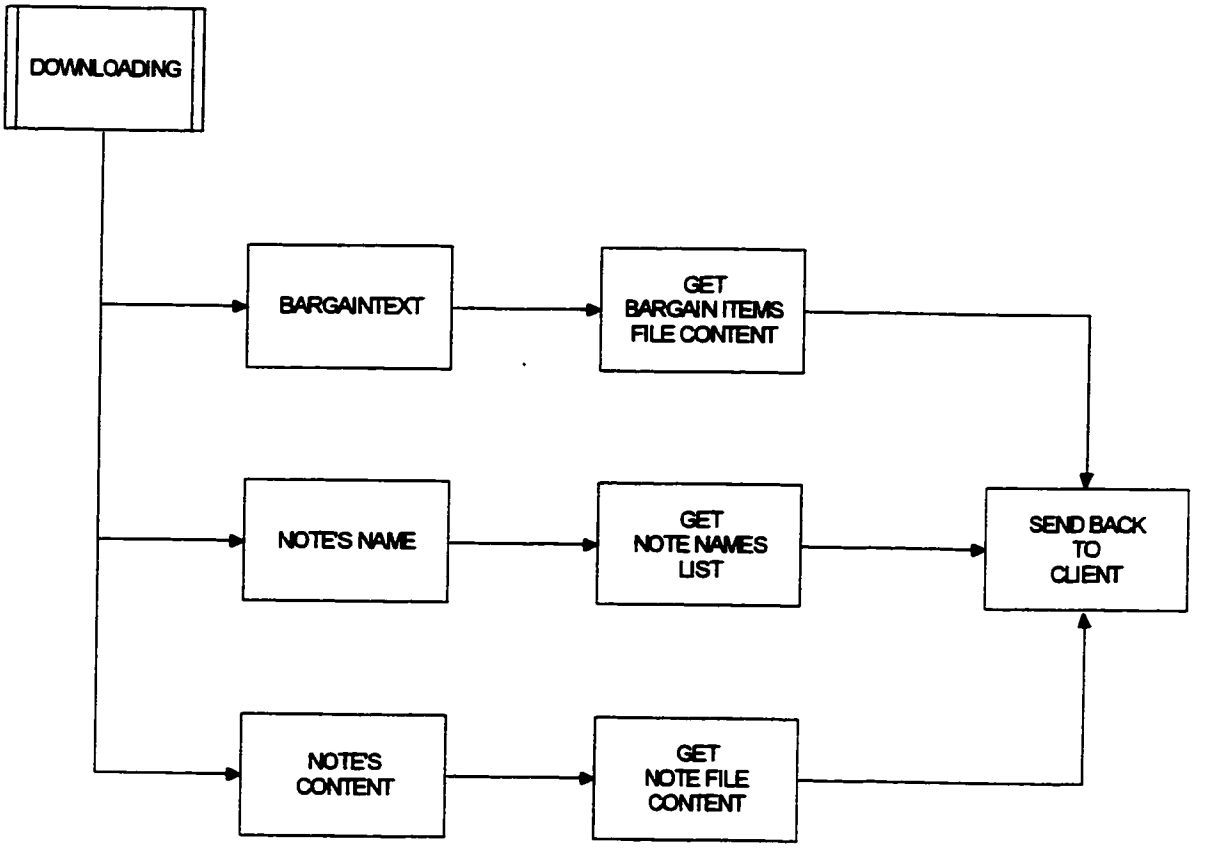
A7. Open Session Windows Flowchart (cont)



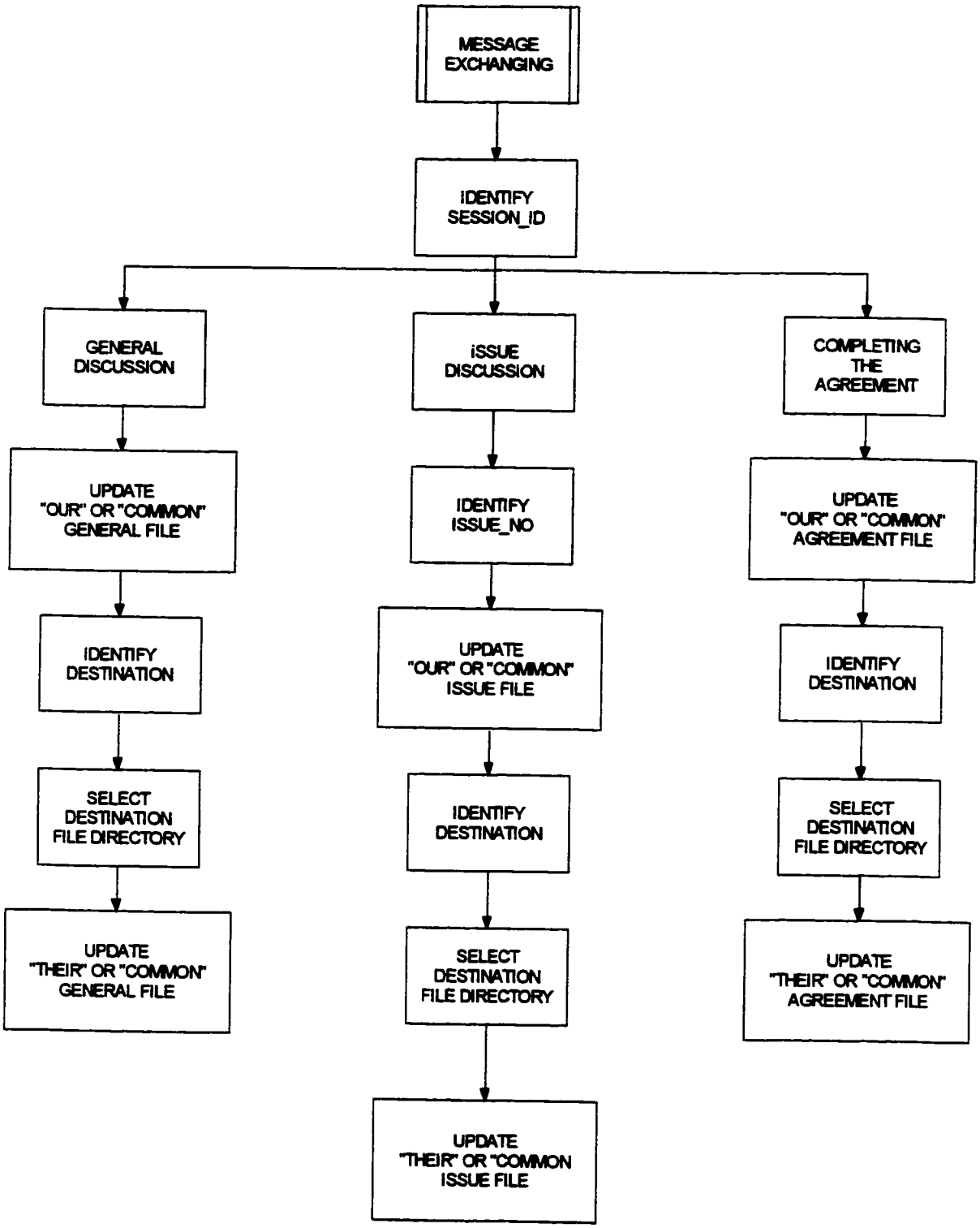
A8. Server System Flowchart



A8. Server System Flowchart (cont)



A8. Server System Flowchart (cont)



A8. Server System Flowchart (cont)

Appendix B. Java Language Elements

A Java program consists of at least one file with an extension **.java** that holds the Java source code. This file is called a *compilation unit*. Each compilation unit consists of at least one of four elements: package statements, import statements, class declarations, and interface declarations. The class declarations is composed of instance variables and methods, along with static variables and methods. The structure of a simple Java code is shown as follows:

```
// import the required class from the library
import library-class-name;
// define the class that you want to develop
class Class-Name {
        public static void main (String args[]) {
        // write your code here
        }
}
```

The code that performs the required function is composed of Java tokens which are the basic elements of Java language, such as: comments, identifiers, keywords, literals, operands, and separators.

Comments: To write a comment in a program, Java provides: // comment or /*comment*/.

Identifiers: Identifiers are the name given to variables, classes, and methods to identify them during the compilation. All identifiers must begin with a letter, underscore character, or dollar sign. For example: network_address, comm_port, and number_of_users are valid identifiers.

Keywords: Keywords are identifiers reserved by Java to be used as data type name, command, or instructions. For example: abstract, double, import, package, class, are keywords. The users can not use the keywords as identifiers.

Literals: Literals represent data that the users enter into the program. There are two common literals, numbers and characters. Numbers can be categorized into: integers (decimal, hexadecimal, or octal), floating point, and boolean. Characters can be a single character between single quotes, or an array of characters between double quotes, which is called a String.

Operators and Separators: Operators are characters to be used to perform some type of computation, such as: + to add, - to subtract, * to multiply, and / to divide. Separators are characters to tell the compiler the separation of groups of codes, such as: ; to separate a line of code, { and } to create a block of code.

There are some keywords that define the flow of Java code in a program. These keywords are similar to keywords in C or C++ language.

(1) Perform a loop:

```
for (expression1; expression2; expression3) {
    Statements;
}
```

or

```
do { statements; } while (condition_true);
```

(2) Conditional Statements:

```
if (condition_true) statement1;
else statement2;
```

(3) Select one that satisfy the requirement:

```
switch(variable) {
        case value-1 : statement1; break;
        case value-2 : statement2; break;
        ....
        default: statement;
}
```

Here is how to declare variables in Java:

```
int comm_port = 7576;
String host_name = "mcmail.cis.mcmaster.ca";
boolean connection_on = false;
float pi = 3.142;
int array_of_integer[10];
```

Here is how to declare a method in Java:

```
public void method_name() {
        statements;
}
```

Here is how a class is defined:

```
// define a Circle class
public class Circle {
        public double the_radius, x_center, y_center;
        double pi = 3.14159;
        // the object constructor
        public Circle(double x_center, double y_center, double the_radius) {
                this.x_center = x_center;
                this.y_center = y_center;
                this.the_radius = the_radius;
        }
        // method to calculate circumference
        public double circumference() { return 2*pi*the_radius; }
        // method to calculate area
        public double area() { return pi*the_radius*the_radius;}
}
```

Here is how a circle object is defined:

```
Circle a_circle = new Circle(10, 15, 12);
double circle_area = a_circle.area();
System.out.println("The area of the circle is " + circle_area);
```

Appendix C. Frequently Used Java Functions

Some basic programming techniques used repeatedly in CBSS program are as follows:

(1) Read a text file into a buffer.

```
// define the file and the buffer before using them
String filename = "anyname.txt";
File thefile = new File(filename);
int size = (int) thefile.length();
byte[] thebuffer = new byte[size];
// read the file and put the contents into the buffer
try {
        FileInputStream fis = new FileInputStream(thefile);
        fis.read(thebuffer, 0, size);
        fis.close();
}
catch (IOException e) {System.out.println(e);}
// convert the byte into text and put them in a string buffer (contents)
String contents = new String(thebuffer, 0);
```

(2) Save a text buffer into a text file.

```
// define the file and the buffer
String filename = "anyfile.txt";
File thefile = new File(filename);
String thetext = "....any text .....";
int size = thetext.length();
byte[] thebuffer = new byte[size];
// open the file and save the buffer into it
try {
        FileOutputStream fos = new FileOutputStream(thefile);
        thetext.getBytes(0, size, thebuffer, 0);
        fos.write(thebuffer);
        fos.close();
}
catch (IOException e) {System.out.println(e); }
```

(3) Display a text buffer' s contents in a text window

```
// define the buffer and the window
String buftext;
TextArea textwindow = new TextArea(15,60);
// define the characteristics of the window
textwindow.setFont(new Font("Times Roman", Font.BOLD, 11));
```

```
textwindow.setForeground(Color.black);
textwindow.setBackground(Color.pink);
// put the text buffer into the window
textwindow.appendText(buftext);
// attach the text window into the current window
this.add(textwindow);
```

(4) Read a remote file into a text buffer

```
// define the URL address of the file
URL thefile = new URL("http://pc-suarga-2.business.mcmaster.ca/file.txt");
// define a connection and an input stream
URLConnection theconnect = thefile.openConnection();
DataInputStream dis= new DataInputStream(theconnect.getInputStream());
// define the text buffer and read the file into the buffer
String thebuffer, theline;
thebuffer = dis.readLine();
while ((theline=dis.readLine()) != null) {
        thebuffer += theline;
}
```

(5) Create buttons and display them

```
// define the buttons
Button savefile = new Button("Save this File");
Button posttext = new Button("Post this Window");
Button compose = new Button("Compose a Message");
// create a panel for the buttons
Panel buttonpanel = new Panel();
// put the buttons in the panel
buttonpanel.add(savefile);
buttonpanel.add(posttext);
buttonpanel.add(compose);
// attach the panel into the current window
this.add(buttonpanel);
// display the current window
this.show();
```

(6) Create and display a menu

```
// define the menu strings
String menu1 = "General Discussion";
String menu2 = "Issue Discussion";
String menu3 = "Complete the Agreement";
// create the menu window
Menu session = new Menu("SESSION");
```

```
// attach the menu items to the menu window
session.add(new MenuItem(menu1));
session.add(new MenuItem(menu2));
session.add(new MenuItem(menu3));
// create and set the menu bar
MenuBar  mymenu = new MenuBar();
setMenuBar(mymenu);
```

(7) Respond to user actions.

An interactive system must provide an user-interface such as menu which responds to mouse clicks or a text field where users may type a message, and so on. The system should implement methods to handle user actions.

```
//handle users actions
public boolean handleEvent(Event evt) {
        event_handled = false;
        switch(evt.id) {
                case Event.ACTION_EVENT :
                        // handle mouse clicks on menu items
                        if (((String) evt.arg.equals(menu1)) {
                                try GeneralDisc();
                                catch (IOException e) System.out.println(e);
                                event_handled = true;
                        }
                        else if (((String) evt.arg.equals(menu2)) {
                                try IssueDisc();
                                catch (IOException e) System.out.println(e);
                                event_handled = true;
                        }
                        else if (((String) evt.arg.equals(menu3)) {
                                try Agreement();
                                catch (IOException e) System.out.println(e);
                                event_handled = true;
                        }
                        // handle clicks on  buttons
                        else if (evt.arg = compose) {
                                try ComposeText();
                                catch (IOException e) System.out.println(e);
                                event_handled = true;
                        }
                        else if (evt.arg = savefile) {
                                try SaveFile();
                                catch (IOException e) System.out.println(e);
                                event_handled = true;
                        }
                        else if (evt.arg == posttext) {
                                try PostText();
                                catch (IOException e) System.out.println(e);
```

```
                    event_handled = true;
                }
                break;
        }
        if (event_handled) return true;
        else return super.handleEvent(evt);
}
```

(8) Open a communication channel

The communication channel (port) must be opened first in the Server system, and then the Client listens to that channel.

```
// create a communication channel at the Server site
int port = 4646;
ServerSocket theserver = new ServerSocket(port);
Connection user = new Connection(theserver);

// open the Client channel
// First-define the host and the port
String host = "pc-suarga-2.business.mcmaster.ca";
int port = 4646;
Socket theclient = new Socket(host, port);
// create an input and an output channel for the Client
DataInputStream innet = new DataInputStream(theclient.getInputStream());
PrintStream outnet = new PrintStream(theclient.getOutputStream());
// listen to the channel
StreamListener Client = new StreamListener(innet, outnet);
```

(9) Send a message to the Server

Sending a message to the Server is easy if the communication channel is already opened, the message is printed out to the output channel, for example: outnet.println(message). However, a header must be attached to the message for proper addressing. CBSS uses four tags in the message header which are: session_tag, origin_tag, destination_tag, and the_time.

```
// define the tags
String session_tag = "#01#";
String origin_tag = "union";
String destination_tag = "manag";
String the_time, the_message;
// get the time
Date today = new Date();
int minute = today.getMinutes();
int hours = today.getHours();
Integer mnt = new Integer(minute);
Integer hrs = new Integer(hours);
the_time = hrs.toString() + ":" + mnt.toString();
```

```
// add the header and then send it out
the_message = session_tag + origin_tag + destination_tag + the_time + the_message;
outnet.println(the_message);
```

(10) Receive a message from the Server

```
// read the message from the input channel

the_message = innet.readLine();
// separate the tags from the message
session_tag = the_message.substring(0,4);
origin_tag = the_message.substring(4,9);
destination_tag = the_message.substring(9,14);
the_message = the_message.substring(14);
```

(11) A Simple Server program

```
// import the input-output and network libraries
import java.io.*;
import java.net.*;
// define the Server class
public class Server extends Thread {
        // these are the global data
        public final static int Port = 4646;
        protected int port;
        protected ServerSocket the_channel;

        // create the comm. channel and then start the comunication
        public Server(int port) {
                if (port==0) port = Port;
                this.port = port;
                try { the_channel = new ServerSocket(port); }
                catch (IOException e) { System.out.println("Error " + e); System.exit(1); }
                System.out.println("Server opened channel : " + port);
                this.start();
        }

        // run the Server until users kill the program
        public void run() {
                try {
                        while(true) {
                                        // accept a client then open a connection
                                        Socket the_client = the_channel.accept();
                                        Connection c = new Connection(the_client);
                        }
                }
                catch (IOException e) { System.out.println("Error " + e); }
        }
```

```
        // here is the main program
        public static void main(String[] args) {
                int port = 0;
                // check the command line arguments
                if (args.length == 1) {
                  try { port = Integer.parseInt(args[0]); }
                  catch (NumberFormatException e) port=0;
                }
                new Server(port);
        }
}

// define a class to handle the communications with the clients
class Connection extend Thread {
        protected Socket client;
        protected DataInputStream innet;
        protected PrintStream outnet;

        // initialization
        public Connection(Socket client_socket) {
                client = client_socket;
                try {
                        innet = new DataInputStream(client.getInputStream());
                        outnet = new PrintStream(client.getOutputStream());
                }
                catch (IOException e) {}
                this.start();
        }

        // provide the services
        public void run() {
                String message;
                try {
                        for (;;) {
                                // get the message from a client
                                message = in.readLine();
                                // process the message
                                .....
                                // send back the processed message
                                outnet.println(message);
                        }
                }
                catch (IOException e);
        }
}
```

(12) A Simple Client program

This program is an example of a Client system implementation. The program consists of two

classes of program which run in parallel (multi-threaded). The first class is Client class, to receive a message from a user and to send this message to the Server system. The second class is StreamListener class, to receive a message sent by the Server and to deliver this message to the Client class to be displayed.

```
// import libraries
import java.io.*;
import java.awt.*;
import java.net.*;
import java.applet.*;

// define the Client class
public class Client extends Applet {
        // define the host and the communication port
        static String host = "pc-suarga-2.business.mcmaster.ca";
        static int port = 4646;
        Socket s;
        // define the input and output channel
        DataInputStream innet;
        PrintStream outnet;
        // define an input and an output text area
        TextField user_input;
        TextArea user_output;
        // define listener object to listen to comm channel
        StreamListener listener;

        // initialize the connection
        public void init() {
                try {
                        // create a new communication socket connected
                        // to a host through a certain port
                        s = new Socket(host, port);
                        // create the input and output communication channel on the socket
                        innet = new DataInputStream(s.getInputStream());
                        outnet = new PrintStream(s,getOutputStream());

                        // create the users area
                        user_input = new TextField();
                        user_output = new TextArea();
                        this.add("North", user_input);
                        this.add("Center", user_output);

                        // open the connection
                        listener = new StreamListener(innet, user_output);

                        // inform the user that the connection is opened
                        this.showStatus("You are connected to host" + host);
                }
                catch (IOException e) this.showStatus("Error " + e.toString());
        }
```

```
        // get the message from the user_input area whenever a text is typed
        // and then send it to the Server
        public boolean action(Event e, Object whatever) {
                if (e.target == user_input) {
                    outnet.println((String) e.arg);
                    user_input.setText("");
                    return true;
                }
                return false;
        }
}

// this is the class that handles the communication
class StreamListener extends Thread {
        DataInputStream innet;
        TextArea output;

        public StreamListener(DataInputStream innet, TextArea output) {
                this.innet = innet;
                this.output = output;
                this.start();
        }

        public void run() {
                String message;
                try {
                     for (;;) {
                         message = innet.readLine();
                         if (message == null) break;
                         output.setText(message);
                     }
                }
                catch (IOException e) output.setText(e.toString());
        }
}
```

Appendix D. CBSS Home Page HTML Files.

D.1 INDEX.HTML File:

```
<HTML>
<HEAD>
<TITLE>CBSS!</TITLE>
</HEAD>
<BODY background="bluebackg.gif">
<P>
<table align=left border=0 valign=middle>
<tr>
<td valign=top>
   <table border=0>
   <tr><td><a href="intro.html"><img src="intro.gif"></a></td></tr>
   <tr><td><a href="require.html"><img src="require.gif"></a></td></tr>
   <tr><td><a href="limit.html"><img src="limit.gif"></a></td></tr>
   <tr><td><a href="cbss.html"><img src="start.gif"></a></td></tr>
   </table>
</td>
<td>
 <table border=1>
 <tr><td align=center>
 <img src="wcbss.gif" width=400 height=100>
 </td></tr>
 <tr><td align=center>
    <table border=0>
    <tr>
    <td><img src="wcbss3.gif" width=400 height=175></td>
    </tr>
    </table>
 <tr><td align=center><strong>
  <a href="mailto:suargas@mcmaster.ca">Developed by Suarga</a>
  </strong></td></tr>
  </table>
</td>
</tr>
</table>
</BODY>
</HTML>
```

D.2 INTRO.HTML File.

```
<HTML>
<HEAD>
<TITLE>CBSS!</TITLE>
</HEAD>
<BODY background="bluebackg.gif">
<P>
<table align=left border=0>
<tr>
<td valign=top>
  <table border=0>
  <tr><td><a href="intro.html"><img src="intro.gif"></a></td></tr>
  <tr><td><a href="require.html"><img src="require.gif"></a></td></tr>
  <tr><td><a href="limit.html"><img src="limit.gif"></a></td></tr>
  <tr><td><a href="cbss.html"><img src="start.gif"></a></td></tr>
  </table>
</td>
<td valign=top>
<h2>Web-CBSS</h2>
<p>
Web-CBSS is a Web-based Collective Bargaining Support System. It is a program
written in Java language to support a remote negotiation process between a
Management team and a Union team. A mediator is allowed to join the negotiation
process in case the mediator is needed.
<p>
This program had been tested by students who took a Collective Bargaining class
during the fall term of 1996 and winter term of 1997.
<p>
The program provides four modules:<br>
<p>
(1) <b>Pre-Session :</b> module to prepare information required during the negotiation,
such as: bargaining items, and notes. The menu provides:<br>
a. Type In Bargain Items<br>
b. List the Bargain Items<br>
c. View/Edit Bargain Items<br>
d. Prepare a Note<br>
e. View/Edit a Note<br>
<p>
(2) <b>Session :</b> module that manages the discussion process. The menu provides:<br>
a. General Discussion<br>
b. Issue Discussion<br>
c. Completing the Agreement<br>
<p>
(3) <b>Help :</b> module that provides on-line help. The menu provides:<br>
(a) Help on Pre-Session<br>
(b) Help on Session<br>
```

(c) Help on Hotline
(d) How to Copy & Paste
<p>
(4) HotLine : a window for a short and immediate communication among the teams (terminals).
<p>
To Start the program:
a. Click Start button, wait for a minute while the system is downloading CBSS.
b. Select your team's name, Management, Union, or Mediator. A HotLine window is displayed. You may "minimize" your Netscape windows by clicking the minimize button [-] on the top right corner.
c. Start by clicking the "Pre-Session" menu, and select the appropriate menu items.
d. Click the "Session" menu, to start the General Discussion, Issue Discussion, and Completing the Agreement.
e. To end the program, bring back/up your Netscape window and close it.
</td>
</tr>
</table>
</body>
</html>

D.3 REQUIRE.HTML File.

```
<HTML>
<HEAD>
<TITLE>CBSS!</TITLE>
</HEAD>
<BODY background="bluebackg.gif">
<P>
<table align=left border=0>
<tr>
<td valign=top>
  <table border=0>
  <tr><td><a href="intro.html"><img src="intro.gif"></a></td></tr>
  <tr><td><a href="require.html"><img src="require.gif"></a></td></tr>
  <tr><td><a href="limit.html"><img src="limit.gif"></a></td></tr>
  <tr><td><a href="cbss.html"><img src="start.gif"></a></td></tr>
  </table>
</td>
<td>
<h2>Requirements</h2>
<p>
This program is written in Java language. A computer with a multitasking
operating system such as Windows NT, Windows 95, and Unix is required
```

```
to run this program.
<p>
The Web browser should be a Java capable browser. There are some Java
instructions do not work well under Internet Explorer (IE) version 3.0.
The best view can be seen by using Netscape 3.0 or Netscape Gold 3.0.
<p>
A Java Applet is presented in the following, if your browser can display it
your system is capable to run CBSS program, otherwise your system is unable
to run CBSS program.
<p><center>
<APPLET CODE="gambar.class" width=408 height=44>
<PARAM name = "messages" value = "Welcome .....!;Web Based CBSS;Collective Bargaining
Support System">
<PARAM name = "msgcolor" value = "(150,255,50),(255,150,50),(50,255,150)">
<PARAM name = "control" value = "0,1,5,0,1,4,0,3,5">
<h3>WELCOME.... TO Web Based CBSS----COllective Bargaining Support System on then
Web</h3>
A JAVA applet should be shown here to WELCOME you,
but your browser is unable to display a JAVA Applet.
</APPLET>
</center>

</td>
</tr>
</table>
</body>
</html>
```

D.4 LIMIT.HTML File.

```
<HTML>
<HEAD>
<TITLE>CBSS!</TITLE>
</HEAD>
<BODY background="bluebackg.gif">
<P>
<table align=left border=0>
<tr>
<td valign=top>
  <table border=0>
  <tr><td><a href="intro.html"><img src="intro.gif"></a></td></tr>
  <tr><td><a href="require.html"><img src="require.gif"></a></td></tr>
  <tr><td><a href="limit.html"><img src="limit.gif"></a></td></tr>
  <tr><td><a href="cbss.html"><img src="start.gif"></a></td></tr>
```

```
   </table>
</td>
<td>
<h2>Limitation</h2>
<p>
This program is under construction, there are some limitations of the
current version.
<p>
(1) Only one virtual discussion room is available, additional rooms can be added on request.<br>
(2) Only three teams are allowed in the room.<br>
(3) Automatic initialization for a new negotiation process has<br>
   not been implemented yet.<br>
(4) Local file access is not allowed.<br>
(5) Maximum issue items to be discussed is 9. <br>
(6) Maximum number of notes allowed is 10.<br>
</td>
</tr>
</table>
</body>
</html>
```

D.5 CBSS.HTML File.

```
<HTML>
<HEAD>
<TITLE>The Program</TITLE>
</HEAD>
<BODY background="bluebackg.gif">
<p>
<strong>
Please "Minimize" this window <br>
for the best view whenever you <br>
are ready to start the discussion.
<p>
To end the discussion, bring this <br>
window back from the status bar<br>
and then "Close" it.
<P>
<APPLET CODE="cbss.class" WIDTH=5 HEIGHT=5>
</APPLET>
</BODY>
</HTML>
```

Appendix E. Collective Bargaining Simulation (Experiment #1)

General Background Information

The Ontarioville Municipal Collective Bargaining Simulation

Ontarioville is a small but growing city (pop. 50 000) near Oakville, Ontario. The city currently employs 1 000 workers. Of these, 200 are full-time outside employees. The duties of the outside workers include public park and winter ice rink maintenance, snow plowing, and garbage collection (among others). It is an exciting time for the outside employees because they have just become certified as the city' s first municipal union, CUPE Local 1498.

It is now time for the two parties to go to the bargaining table. There is a great deal of uncertainty over what will happen during the negotiations, partly because the process and outcomes of first-contract negotiations have important implications for the future bargaining relationship. Because the former NDP Government' s Social Contract legislation (which expired March 31, 1996), employees have been without a pay increase for nearly four years. Recent cutbacks in transfer payments by the Harris' Government has exacerbated the city' s budget situation. The city is pessimistic about future revenue projections. The citizens of Ontarioville have formed a Taxpayers Coalition in response to what they feel are needlessly high property taxes, and they have made it clear that they will not tolerate another increase in assessments in the next year.

The unemployment level in Ontarioville has closely paralleled the provincial average since 1990.

Negotiations (First simulation)

While a number of issues have to be resolved through formal negotiations, the parties have already agreed upon a number of items. Some of them are: union recognition, management rights, job classifications, hours of work and breaks, seniority, grievance and arbitration procedures, holidays, uniforms, and maintaining existing fringe benefits.

Management and the Union agreed to submit their positions on the remaining items by mail. This has been done and the following is a synopsis of the demands (you may assume that all matters previously agreed upon are not to be subject of negotiations):

Union Demands

1. Wage parity with a comparable bargaining unit in a nearby city called Oakville:

Classification	Ontarioville Current Rate*	Oakville Current Rate
Job Group A (50 employees)	$ 11.88	$ 14.86
Job Group B (75 employees)	$ 12.70	$ 15.58
Job Group C (50 employees)	$ 12.96	$ 16.20
Job Group D (25 employees)	$ 13.62	$ 17.03

*NOTE: For the purpose of negotiations, there are 1875 hours in one work Year (37.5) hours per week * 50 weeks per year).

2. A one-year contract.

3. A "call-in" provision: an employee called back to work shall be guaranteed a minimum of four hours of paid work and all call-in hours shall be paid at overtime rates.

4. Union security: a union shop.

Company Demands/Offer

1. A wage rollback of five percent, followed by a freeze for the remainder of the collective agreement.

2. Contract duration of 3 years.

3. The employer proposes that employees be paid only for the call-in hours they work. Overtime entitlement will not accrue unless employees work in excess of the hours specified in the Employment Standard Act.

4. The employer opposes a union shop.

Negotiation (Second simulation)

The first collective agreement between the parties has expired. All of the major economic issues have been settled. However, the union is seeking to address a number of major concerns that it did not push hard in the interest of achieving a basic first collective agreement. For its part, the employer continues to trim costs as it struggles under cuts in provincial transfer payments and its inability to generate new revenue sources. The following issues have not been resolved. There is no need to cost the second collective agreement.

Union Demands

1. Seniority: A sufficient ability clause for all promotions and transfers.
2. Layoffs: The current agreement does not deal with layoffs and the union indicates it will submit language for a layoff procedure that provides comprehensive job security protection.
3. A union representation clause that provides that members of the union negotiating committee will be paid for time spent in negotiations during regularly scheduled working hours.

Employer Demands

1. Seniority: maintain the current relative ability clause for promotions and transfers.
2. Layoffs: The employer opposes any restrictions on its right to determine the size of the workforce.
3. The union shall reimburse the employer for all time paid to members of the negotiating committee while not at work.

Appendix F. Collective Bargaining Simulation (Experiment #2)

General Background Information

The Ontarioville Municipal Collective Bargaining Simulation

Ontarioville is a small but growing city (pop. 50 000) near Oakville, Ontario. The city currently employs 1 000 workers. Of these, 200 are full-time outside employees. The duties of the outside workers include public park and winter ice rink maintenance, snow plowing, and garbage collection (among others). It is an exciting time for the outside employees because they have just become certified as the city's first municipal union, CUPE Local 1498.

It is now time for the two parties to go to the bargaining table. There is a great deal of uncertainty over what will happen during the negotiations, partly because the process and outcomes of first-contract negotiations have important implications for the future bargaining relationship. Because the former NDP Government's Social Contract legislation (which expired March 31, 1996), employees have been without a pay increase for nearly four years. Recent cutbacks in transfer payments by the Harris' Government has exacerbated the city's budget situation. The city is pessimistic about future revenue projections. The citizens of Ontarioville have formed a Taxpayers Coalition in response to what they feel are needlessly high property taxes, and they have made it clear that they will not tolerate another increase in assessments in the next year.

The unemployment level in Ontarioville has closely paralleled the provincial average since 1990.

Negotiations

While a number of issues have to be resolved through formal negotiations, the parties have already agreed upon a number of items. Some of them are: union recognition, management rights, job classifications, hours of work and breaks, seniority, grievance and arbitration procedures, holidays,

uniforms, and maintaining existing fringe benefits.

Management and the Union agreed to submit their positions on the remaining items by mail. This has been done and the following is a synopsis of the demands (you may assume that all matters previously agreed upon are not to be subject of negotiations):

Union Demands

1. Wage parity with a comparable bargaining unit in a nearby city called Oakville:

Classification	Ontarioville Current Rate*	Oakville Current Rate
Job Group A (50 employees)	$ 11.88	$ 14.86
Job Group B (75 employees)	$ 12.70	$ 15.58
Job Group C (50 employees)	$ 12.96	$ 16.20
Job Group D (25 employees)	$ 13.62	$ 17.03

*NOTE: For the purpose of negotiations, there are 1875 hours in one work Year (37.5) hours per week * 50 weeks per year).

2. A short contract.
3. Job security: comprehensive language regulating contracting out and layoffs generally.
4. A "call-in" provision with a minimum guarantee on hours at overtime rates.
5. Annual sick leave of 18 days (1 and ½days per month), with the right to accumulate unused days up to a maximum of 90 days.
6. A union representation clause that provides members of the union negotiating committee will be paid for the time spent in negotiations during regularly scheduled working hours.

Company Demands/Offer

1. A wage rollback of five percent, followed by a freeze for the remainder of the collective agreement.
2. Contract duration of 3 years.
3. Opposes any job security provisions impinging on management flexibility.

4. The employer proposes that employees be paid only for the call-in hours they work and overtime entitlement will not accrue unless employees work in excess of the hours specified in the Employment Standard Act.

5. Status quo which is 6 sick days per year and no accumulation of sick days. The absenteeism rate has increased in recent years wand was 5.8 days per employee in calendar year 1996.

6. The union shall reimburse the employer for all time paid to members of the negotiating committee while not at work.

APPENDIX G. Questionnaires

CBSS QUESTIONNAIRE #1

Please feel free to express your judgment on CBSS by evaluating the following statements with a number between 1 and 5, where:

1=strongly disagree,
2=disagree,
3=neutral,
4=agree,
5=strongly agree.

(A) As a support system to collective bargaining process, CBSS:

1 2 3 4 5 (1) helped me to focus more on the issue problem than on the negotiators' personality.

1 2 3 4 5 (2) helped me in reducing my hostility to the other side' s negotiators.

1 2 3 4 5 (3) gave me more time to think before I contributed my ideas into the discussion.

1 2 3 4 5 (4) helped me in understanding the other side' s negotiators concerns and problems.

1 2 3 4 5 (5) helped me in preparing the final contract.

1 2 3 4 5 (6) had valuable contributions to the negotiation outcomes.

(B) I feel that CBSS:

1 2 3 4 5 (7) sped up the negotiation process.

1 2 3 4 5 (8) made the negotiation process easy to organize

1 2 3 4 5 (9) improved the communication between two sides

1 2 3 4 5 (10) reduced my ability to express my ideas clearly

(C) As a computer software for Collective Bargaining, I think that CBSS:

1 2 3 4 5 (11) is a useful tool for negotiators.

1 2 3 4 5 (12) is a user-friendly computer software.

1 2 3 4 5 (13) may be used for real bargaining situations.

1 2 3 4 5 (14) is a kind of software that I avoid using.

(D) Please give your own evaluations and comments on CBSS in the following space

(E) The negotiation was started at: ..... and finished at: .....

(F) I finished negotiating ..... issues out of ..... issues during that time.

Thank you very much for your participation in this experiment.

CBSS QUESTIONNAIRE #2

Please feel free to express your feeling about CBSS by answering the following questions with a number between 1 and 5, where:

1=strongly disagree, 2=disagree, 3=neutral,
4=agree, 5=strongly agree.

(A) Compared to the traditional bargaining process (FTF), the process supported by CBSS:

1 2 3 4 5 (1) was a more successful process.

1 2 3 4 5 (2) was better in attaining my team' s goals.

1 2 3 4 5 (3) gave me a more satisfying contract outcome.

1 2 3 4 5 (4) was more efficient in time usage.

1 2 3 4 5 (5) was more effective (i.e we were able to make better decisions).

1 2 3 4 5 (6) was easier because of this software.

1 2 3 4 5 (7) was faster because of this software.

1 2 3 4 5 (8) was better overall because of this software.

1 2 3 4 5 (9) was more relaxed.

1 2 3 4 5 (10) was more stressful.

(B) Compared to the traditional bargaining process, the electronic communication supports of CBSS to the bargaining process:

1 2 3 4 5 (11) improved the communication between the negotiating parties.

1 2 3 4 5 (12) slowed down the negotiation process.

1 2 3 4 5 (13) blocked the communication between the parties involved.

(C) Compared to the traditional bargaining process, CBSS structures the bargaining process into steps. This structured process of CBSS:

1 2 3 4 5 (14) helped me to focus more on the issues than on personalities of the other party members.

1 2 3 4 5 (15) let me think before answering or replying to other side' s comments.

1 2 3 4 5 (16) made the process slower than the traditional process.

1 2 3 4 5 (17) made the negotiation easier than the traditional process.

(D) If I have choice:

1 2 3 4 5 (18) I **prefer** to use CBSS to negotiate **rather than** a face-to-face meeting.

1 2 3 4 5 (19) I **prefer** to use CBSS **combined** with a face-to-face meeting.

1 2 3 4 5 (20) I will use CBSS **IF** face-to-face meeting is not possible.

(E) Please give your own evaluations and comments on CBSS in the following space

(F) The negotiation was started at: ..... and finished at: .....

(G) I finished negotiating ..... issues out of ..... issues during that time.

Thank you very much for your participation in this experiment.

Web-based Collective Bargaining Support System Questionnaire #3.

Please respond to the following questions: (you may use additional papers if necessary, or you may type your response using wordprocessor).

(1) As a system to support negotiation process, does CBSS offer advantages over face-to-face negotiations? What are the advantages?

(2) What are the disadvantages of using a system like CBSS in conducting negotiations?

(3) Do you think that CBSS can be used as an alternative to conducting a negotiations if a face-to-face meetings are not possible? What are the possible negotiation settings where CBSS can be used?

(4) Do you think it is possible to use CBSS in conjunction with face-to-face negotiations? Under what conditions?

(5) Would you prefer to use CBSS rather than bargaining face-to-face? Why or why not?

(6) Do you think that it is possible to get better outcomes using CBSS rather than face-to-face negotiations? Why or why not?

(7) Collective bargaining is often a time consuming process. Would the use of CBSS as a support system increase or decrease the length of bargaining process? Explain your response.

Glossary

3GL (third generation language) Traditional (procedural) programming language like COBOL and FORTRAN.

API (application programming interface) Specification or actual function library that provides a standard look to programs. For example, JAVA provides many APIs that are ready to use.

Applet A Java program that can be embedded into a Web-page.

Back end A term that sometimes used instead of the term Server to indicate the data source or the system that provides services.

Backbone The central transmission path in a network.

bit (binary digit) The smallest unit of information used or processed by the computer, it is either 0 or 1.

Browser A computer software for surfing the Web. It provides facilities to view multimedia contents (text, image, video, audio) and to move from one Web-site to other Web-sites. The most popular browser are Netscape and Internet Explorer (IE).

Buffer Temporary storage area. Similar to a scratch pad.

byte The unit of memory and disk-storage capacity, 1 byte = 8 bit.

C/C++ C is a traditional procedural 3GL, but C++ is an object-oriented language that happens to have been built on C.

CBSS (collective bargaining support system) A Web-based software designed to support collective bargaining process or negotiation process in general by providing electronic communication supports over the Internet, process structuring, and discussion documentation.

Class In object-oriented terminology, an abstract definition of a group of objects that have similar characteristics, including associated code and data structures. Individual instances of a class are objects.

Client A computer or a program in a computer that request services from a server such as printing, information retrieval, data updating, etc.

CPU (central processing units) The "brains" of a computer, a processor, or a microprocessor. Popular CPUs include Intel 80x86, Intel Pentium, RISC chips.

C/S (Client/Server) It is an architecture, or a modular way of designing information system such that the workload is divided between client computers that request services and the server computer that process the request.

DBMS (database management system) Software that lets you create and maintain a database. Examples range from desktop DBMSs like Paradox and dBase to enterprise DBMSs like Oracle and SQL Server.

DSS (decision support system) An interactive computer-based system that provides a decision maker with easy access to decision models and data in order to support semi-structured and ill-structured decision making tasks.

E-mail (electronic mail) Messages themselves or the software controlling their creation and routing through computer networks such as Internet.

Ethernet Popular 10 Megabit/second (Mbps) LAN protocol. It supports packet size ranging from minimum of 64 bytes to 1.5 Kilobytes.

Front end A term used as a synonym of Client to refer to the program running on a PC.

FTP (file transfer protocol) Popular Internet protocol for transferring (downloading) file from one computer (the host) to another.

GDSS (group decision support system) An interactive computer-based system which facilitates solution of unstructured problems by a set of decision makers working together as a group.

Giga 1 Giga = 1000 million = 1000 Mega = 1,000,000 Kilo = 1,000,000,000. Example: 1 Gigabyte (GB) ≈ 1,000 Megabyte (MB) ≈ 1,000,000 Kilobyte (KB) ≈ 1000,000,000 bytes.

Gopher A server protocol which provides the clients an easy navigation of information resources (text-based) over the Internet by using a menu of text-labeled choices..

GUI (graphical user interface) User interface which uses graphics, icons, and windows, the standard used by operating system such as Microsoft Windows, X Windows, Motif, or Macintosh desktop. Compare to character-based interface used by MS-DOS.

HTML (hypertext markup language) It is a language consists of tags used to describe the structure of a document and hyperlinking information. A text file embedded with

HTML tags is called HTML file. A Web browser reads the HTML file and interprets the tags and then displays the document accordingly.

HTTP (hypertext transfer protocol) Protocol used by Web servers to distribute HTML files to clients who request the files.

Hyperlink Highlighted object in a document which provides link to other media such as documents, phrases, movie clips, or audio clips. The link is not limited to only same Web-server but to other servers anywhere in the world.

Internet An international network of telecommunications channels that links individuals to source of data and permits information exchanges. It is the largest interconnected computer networks which communicate using TCP/IP.

JVM (Java virtual machine) A collection of Java interpreter and run-time system which interprets and executes programs written in Java language.

Kilo 1 Kilo = 1000. For example, 1 Kilobit (Kb) ≈ 1000 bit, 1 Kilobyte (KB)≈1000 byte (note: 1 Kilobyte = 1024 byte).

LAN (local area network) A network of computers in a local proximity, usually for sharing resources (files, peripherals) and exchanging messages.

MAN (metropolitan area network) A public high speed network, operating at 100 megabits per second, or faster, capable of voice and data transmission over a distance of up to 80 kilometers. A MAN is smaller than a WAN but larger than a LAN.

Mega 1 Mega = 1,000 Kilo = 1 million. For example, 1 Megabyte ≈ 1 million bytes.

Middleware A component of a C/S system which sits between the client and the server. There are some categories of middleware such as: network, messaging, or database middleware.

Multi-thread An ability of an operating system, or a program to execute more than one task at the same time.

NSS (negotiation support system) A system consisting of hardware, software, people, procedures, and data that assist the individual negotiators, negotiation team, and third party. NSS advices, provides a solution, or facilitates the process of negotiation.

Open System A term to describe systems which are interoperable with other systems. Compare to proprietary systems which are operable on the same systems.

Server The component of C/S system which provides services or process requests.

SQL (structured query language) A de facto standard query language for relational databases.

TCP/IP (Transmission Control Protocol / Internet Protocol) Communication protocols that encompasses media access, packet transport, session communications, file transfer, e-mail, and terminal emulation. TCP/IP is supported by a large number of hardware and software vendors and is available on many different computers, from PCS to mainframes.

URL (uniform resource locator) A term given to an Internet address of a Web page. URL contains information about the access method to use (which is http:) and the resource to be accessed (for example //www.netscape.com).

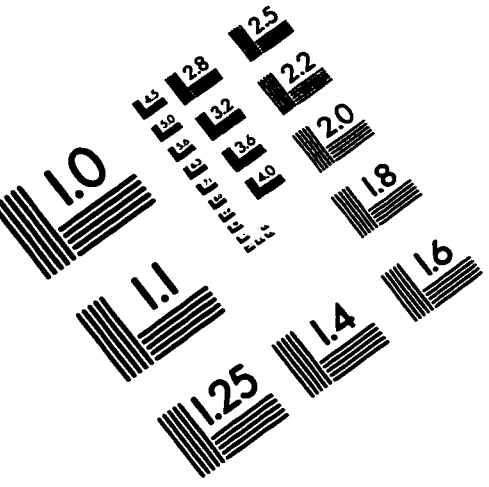
Virtual Memory A memory management technique that allows information in physical memory to be swapped out to a hard disk if necessary. This technique provides applications with more memory space than is actually available in the computer.

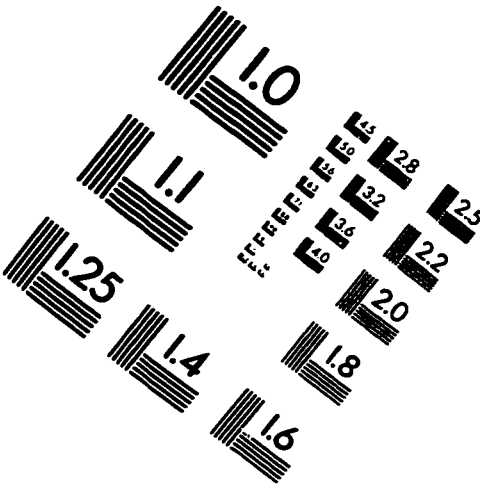
WAN (wide area network) A computer network that connects users across large distances, often crossing the geographical boundaries of cities or states.

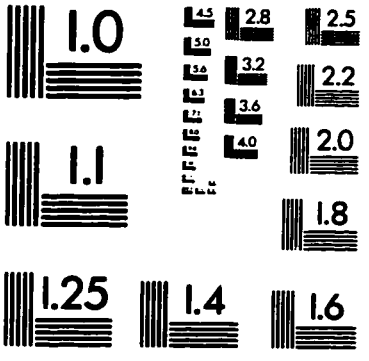
WISIWYS (what I see is what you see) A term given to a user interface that provides the same views for all users.

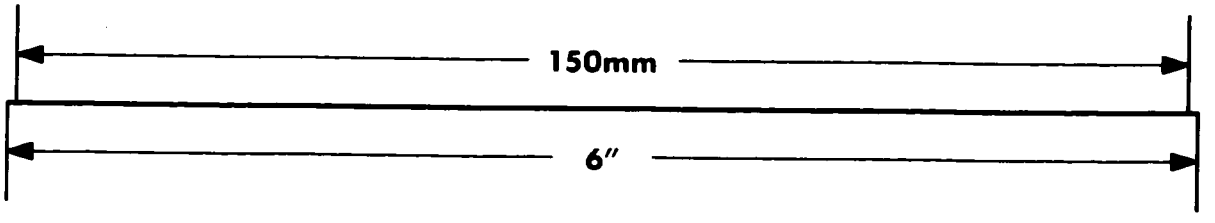
WWW (world wide web) It is also known as W3 or the Web, an Internet service which contains a huge collection of hypertext pages. Hypertext links connect pieces of information (text, graphics, audio, or video) in separate HTML pages located at the same or different Internet sites, and the users explore these pages and links using a Web browser. Users can also access Web resources by specifying appropriate URL of an Internet site.

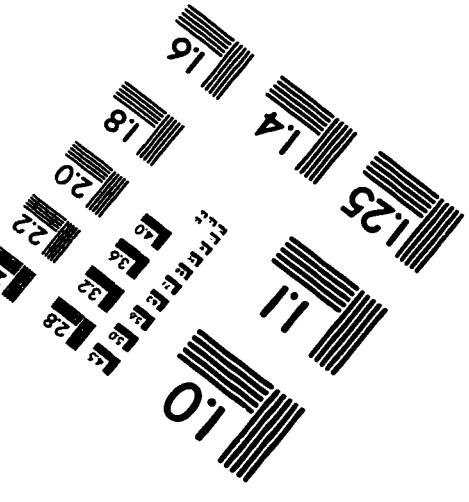
IMAGE EVALUATION TEST TARGET (QA-3)

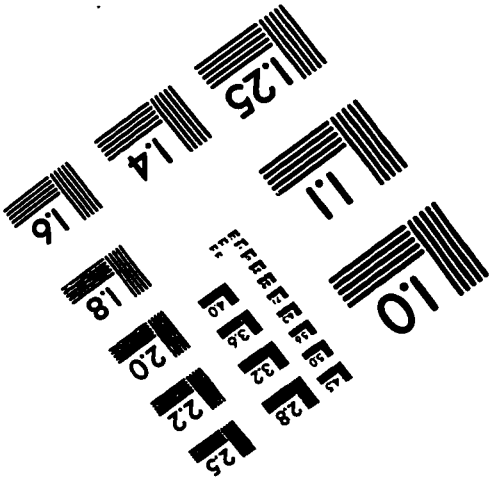












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989