Three-Tier Feature-based Collaborative Browsing for Computer Telephony Integration

HO Ho-ching

A Thesis Submitted in Partial Fulfilment

of the Requirements for the Degree of

Master of Philosophy

in

Computer Science & Engineering

June 2001


香港中文大學
圖書館系統
24 APR 2002
UNIVERSITY
LIBRARY SYSTEM
THE CHINESE UNIVERSITY OF

Three-Tier Feature-based Collaborative Browsing for Computer Telephony Integration

submitted by

Ho Ho-ching

for the degree of Master of Philosophy
at the Chinese University of Hong Kong

Abstract

Traditional call centers provide customers with phone-based service query for Customer Relationship Management (CRM) in business companies. However, this type of customer support service is monotonic as only voice communication is allowed between a customer and a call center operator, who answers the customer's queries. With the emergence of Computer Telephony Integration (CTI), a digital call center, which consists of three critical features — Automatic Call Distributor, Voice-over-IP and Collaborative Browsing, is proposed and prototyped. In contrast to the voice-only communication in traditional call centers, Collaborative Browsing extends the customer-operator communication by supporting collaborative data manipulation through web browsers. A customer not only talks to the operator using Internet based Voice-over-IP technology, but also visually and interactively shares the same data, such as web pages, with a remote operator.

There are several specific features of Collaborative Browsing. Firstly, it is feature-based data synchronization, instead of generic bitmap screen transfer. This is achieved by adopting a Shareable Document Object Architecture (SDOA) to manipulate web contents collaboratively inside a browser. Thus, very low bandwidth is consumed during communication. Secondly, the entire collaboration mechanism is

web oriented so that all data collaboration is based on a common web browser, instead of proprietary application programs that lack functional extensibility. Thirdly, the concept of Hinting is introduced to relate web forms to the backend database schema for identification of the hidden data relationships among form fields. The collaboration process can then be enhanced as an operator obtains additional hints to help a customer manipulate the shared content in a web form. Lastly, the Collaborative Browsing is Computer Telephony (CT) specific. This means the operator-assisted collaboration can be applied to other call center scenarios, such as Interactive Voice Response (IVR) phone services, in additional to the typical PC-to-PC collaboration. Consequently, a customer can manipulate the complicated IVR services more easily with the collaborative assistance of an operator.

To support our designs, two engineering prototypes have been implemented. The first one is a PC-to-PC collaborative browsing system. All the collaborative services, such as tele-pointing, whiteboard, are realized by the SDOA under a three-tier architecture. The core focus is on the hinting mechanism specially designed for collaborative form manipulation to reduce the complexity when handling the related form data during collaboration.

The second prototype is the collaborative IVR system. Its critical contribution lies on extending the form interaction mechanism to a CT-specific Telephone Form, with respect to their architectural similarities. Using Telephone Form, the traditional self-served IVR services can be revolutionized into collaborative form based interactions, where the backend system architecture and the hinting mechanism of the first prototype can be re-used to support data collaboration.

System fine-tuning and performance issues are analyzed in terms of collaboration responsiveness and bandwidth efficiency. Finally, conclusions are drawn based on the engineering experience of the overall system architecture and the technological comparisons with other similar academic and industrial prototypes.

論文題目　：有關電腦電話合成和特徵爲本之三層共同瀏覽系統

作者　：何浩正

學院　：香港中文大學

學部　：計算機科學與工程

修讀學位　：哲學碩士

摘要

在客戶關係管理（Customer Relationship Management）中，傳統的呼叫中心爲客戶提供電話諮詢服務。但是，因爲客戶和遠端服務員只能用語音通話，這種客戶支援服務是單調的。基於電腦電話合成技術（Computer Telephony Integration），我們設計了一個包括自動來電分配器、互聯網語音、共同瀏覽的數字化呼叫中心。共同瀏覽特別能加強客戶和遠端服務員之間的溝通。客戶和遠端服務員可互動地透過視覺來共同瀏覽來共享互聯網上的資源，溝通並不局限於聲音上。

總括而言，共同瀏覽具有幾項特點：首先，它是建基於特徵資料的傳輸，並非點陣式的畫面傳遞。這是透過一個共享文件物件架構（Shareable Document object Architecture）來處理網上的資料，大減頻寬的需求。第二，所有資料的共享機制都是以互聯網爲本，建構在瀏覽器之上，不須專屬應用程式，易於功能擴充。第三，這個共同瀏覽系統使用了一種『提示』（Hinting）的概念，有助抽取網上表格中的相關資料,方便遠端服務員去協助客戶填寫網上表格。第四，由於這個系統是針對

電腦電話（Ｃｏｍｐｕｔｅｒ　Ｔｅｌｅｐｈｏｎｙ）方面的技術，這使得整個系統可應用在呼叫中心的其他電訊組件上，如互動話音回應（Ｉｎｔｅｒａｃｔｉｖｅ　Ｖｏｉｃｅ　Ｒｅｓｐｏｎｓｅ）。因此，在遠端服務員的共同協助下，客戶可容易地進行繁瑣的互動話音回應電話諮詢服務 。

基於以上之特點，我們架設了兩個供研究用途的工程原型：第一個是電腦對電腦再加上ＶｏＩＰ技術的共同瀏覽系統。所有系統服務，如遠端點示、共享畫板，皆透過該共享文件物件架構和一個三層網絡架構而起作用。這系統之主要重點在於其『提示』機制是特別針對客戶在填寫網上表格時的繁瑣程序 。

第二個原型是協作性互動話音回應（Ｃｏｌｌａｂｏｒａｔｉｖｅ　Ｉｎｔｅｒａｃｔｉｖｅ　Ｖｏｉｃｅ　Ｒｅｓｐｏｎｓｅ）系統，它延伸了網上表格的互動機制至電話表格的概念。透過使用電話表格，一般性的客戶自助互動話音回應系統，可重用第一個原型的軟件架構和其『提示』機制，而被改進爲能支援與遠端服務員共同協作的表格互動機制 。

在闡述了整個系統的軟件架構及架設後，我們會解析整個系統的微調和性能分析，包括系統反應、頻寬之使用效率等。最後，我們根據整個系統建構的工程經驗及對其他同類科研產物的比較作出最後的總結。

Acknowledgment

I would like to thank my supervisor, Prof. Yiu-Sang Moon, who entrusts me to work on this research topic and supports me with much guidance and references along the research period. Moreover, my two colleagues, Mr. Cheung-Chi Leung and Mr. Ka-Nang Yuen have worked closely with me for the past two years in constructing the entire project prototype. Their efforts in developing the Automatic Call Distributor and the Voice-over-IP Communication System are highly related to the Collaborative Browsing System and crucial to the entire project.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Introduction to PBX based Call Center

Traditional call center for Customer Relationship Management (CRM) [1] is solely based on analogue Private Branch Exchange (PBX) [2], which can be regarded as a telephony switchboard system. Such a PBX system is often used in organizations for setting up call centers so that a fixed number of analogue telephony lines can be shared by a relatively larger amount of users for cost effectiveness. Each call center staff is equipped with a telephone terminal in such a way that each staff can dial out for tele-marketing [3] or receive incoming calls for customer support. The customer-staff communication is solely based on traditional telephone calls. This kind of communication for CRM is monotonic as users can communicate with each other in only one dimension (voice). Moreover, the PBX-based call center is difficult to scale up rapidly for larger customer base, with respect to the high cost of purchasing additional telecommunication equipment of the PBX system [4].

Regarding the limitations of the analogue PBX systems in terms of multi-media communication and system reliability [5] and scalability, we would like to revolutionize the analogue call center system into a digital one based on Computer Telephony Integration (CTI) [6], which computerizes the traditional analogue telecommunication technologies. The digital call center system [7], [8] consists of a high-end server as the telephony gateway, instead of a PBX, and a set of PC clients for each call center operator. While the internal network of the call center is an

Intranet, the outside customer calls can be either from the Public Switched Telephone Network (PSTN) or from the Internet.

In particular, multimedia communication can be realized by Collaborative Browsing, which focuses on how data collaboration [9] can be supported between a customer and a call center operator. As a result, the customer and the operator can communicate in more than one dimension. Visual interaction as well as voice chat is allowed in contrast to the traditional voice-only communication. When a customer has problems concerning the shared contents, both the customer and an operator are not confined to oral discussion during a collaborative browsing session but they can also visually perceive and collaboratively work on the shared data space in the web [10].

As a whole, the entire digital call center system is separated into three research areas: Collaborative Browsing, the focus of this thesis, Voice-over-IP studies [11], and Automatic Call Distributor (ACD) [12].

1.2. *The Scenarios*

The digital call center system can be divided into several components: the external Internet and/or PSTN, the call center gateway and the call center Intranet. Inside the call center system, there is a set of the workstations. The work mechanism of the entire system can be illustrated through two scenarios: web-based government taxation and phone banking.

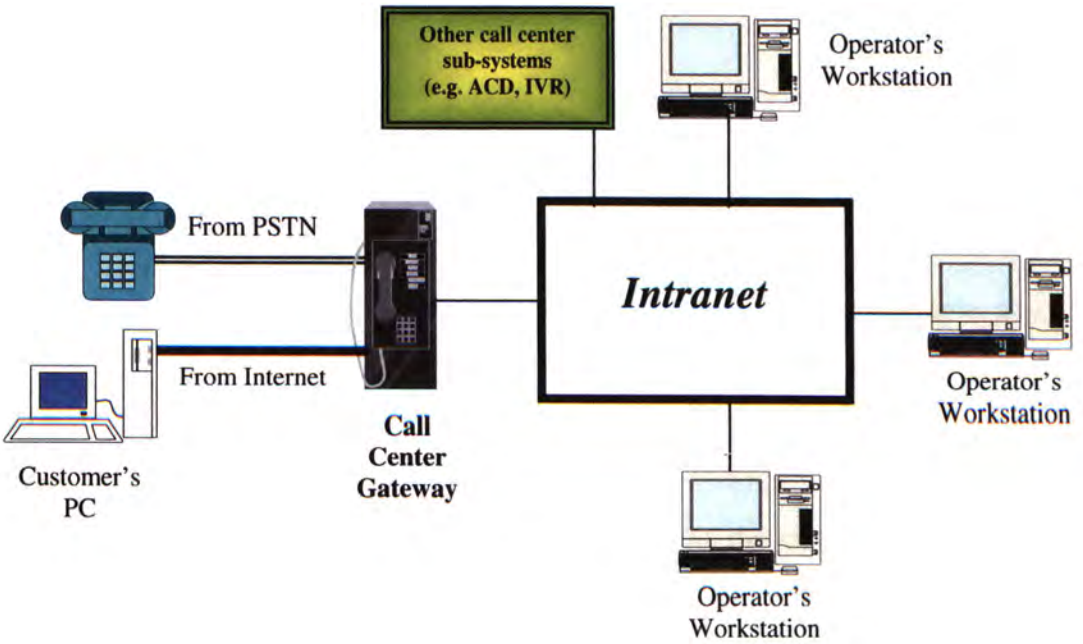


Figure 1 An Example Scenario for the Digital Call Center System

For the case of web-based government taxation, the entire call center system is operated by a group of operators, each of whom controls a workstation in the Intranet. The customers are the citizens, who have to fill in some online tax form for taxation. Suppose that a customer works on a particular tax form at home. Traditionally, when he finds a problem while preparing or reading the form, he can phone to the corresponding government department directly and seek help from the staff there. However, both the staff and the customer only communicate with each other through the phone while the staff cannot perceive the web page content to which the customer is referring.

In our digital call center system, the customer can simply click on a particular button in the browser to request an Internet-based voice communication with a call

center operator. Then, the ACD routes this request to the most suitable operator for the customer. Then, the Internet-based voice communication [13], [14] starts without the need to use a phone.

What is more important is that, with collaborative browsing [15], [16], [17], the operator and the customer can also collaboratively work on the same web contents through their Internet browsers in their PC or workstation. For example, any data input in a tax form on the customer side browser is automatically synchronized to the operator side browser. The operator can also make some visual annotations [18] or data changes on the shared web contents to help the customer tackle his content related problems. This is especially useful if the customer's questions involve some abstract ideas that cannot be elaborated through a voice chat.

Another example is phone banking. Traditionally, phone banking service is realized by a traditional call center, which only supports analogue phone lines through a PBX. A customer phones to the call center for automated self-served banking services, like balance query, through the Interactive Voice Response (IVR) sub-system of the call center. The interaction methodology is that the customer issues requests by pressing keys in the phone keypad and the IVR sub-system responds through the voice output to the customer's phone.

The digital call center can be applied to the above phone banking scenario with the support of collaborative features. In Figure 1, a customer can connect to the digital call center system via a phone. The customer interacts with the IVR sub-system for some phone banking services in the same manner as before. Sometimes,

the customer may have problems in accessing certain services, like pressing the proper set of keys to access his bank account balance. This situation is not rare because the customer can only listen to the voice response from the system. He cannot 'see' the service himself. It is not uncommon that the customer may lose his direction in a complicated IVR tree.

The customer still cannot 'see' the service. However, the operator can record/trace the actions taken by the customer during the IVR using the display of his computer terminal. With such information, the operator is able to have a better understanding of the customer's original desire(s).

Applying the concept of collaborative browsing to IVR, or called collaborative IVR, the customer can press a particular phone key to contact the call center operator so that the operator can assist the customer in accessing a particular service or inputting the correct data into the IVR sub-system.

In general, collaborative IVR is not confined to phone banking, but also other IVR-based phone services, like interactive government services [19], payment-by-phone service [20], voice portal, etc.

1.3. Thesis Overview

Chapter 1 introduces the motivation and the background scenarios. Next, the following three chapters (Chapter 2, 3 and 4) elaborate the three system cores (Basic Collaborative Features, Collaborative Form Manipulation and Collaborative

Interactive Voice Response) in three software engineering aspects: System Design, Architectural Construction and Implementation. Chapter 2 explains the design issues of the collaborative features in the system. After that, the software architecture which shows how different software components interact with each other, is elaborated in Chapter 3. The implementation details are covered in detail in Chapter 4. Chapter 5 discusses the performance testing and optimization on some of the basic collaborative features, which are real time sensitive. Lastly, Chapter 6 concludes our work, makes technical comparisons with some other typical and similar systems and justifies the enabling features of our constructed prototype.

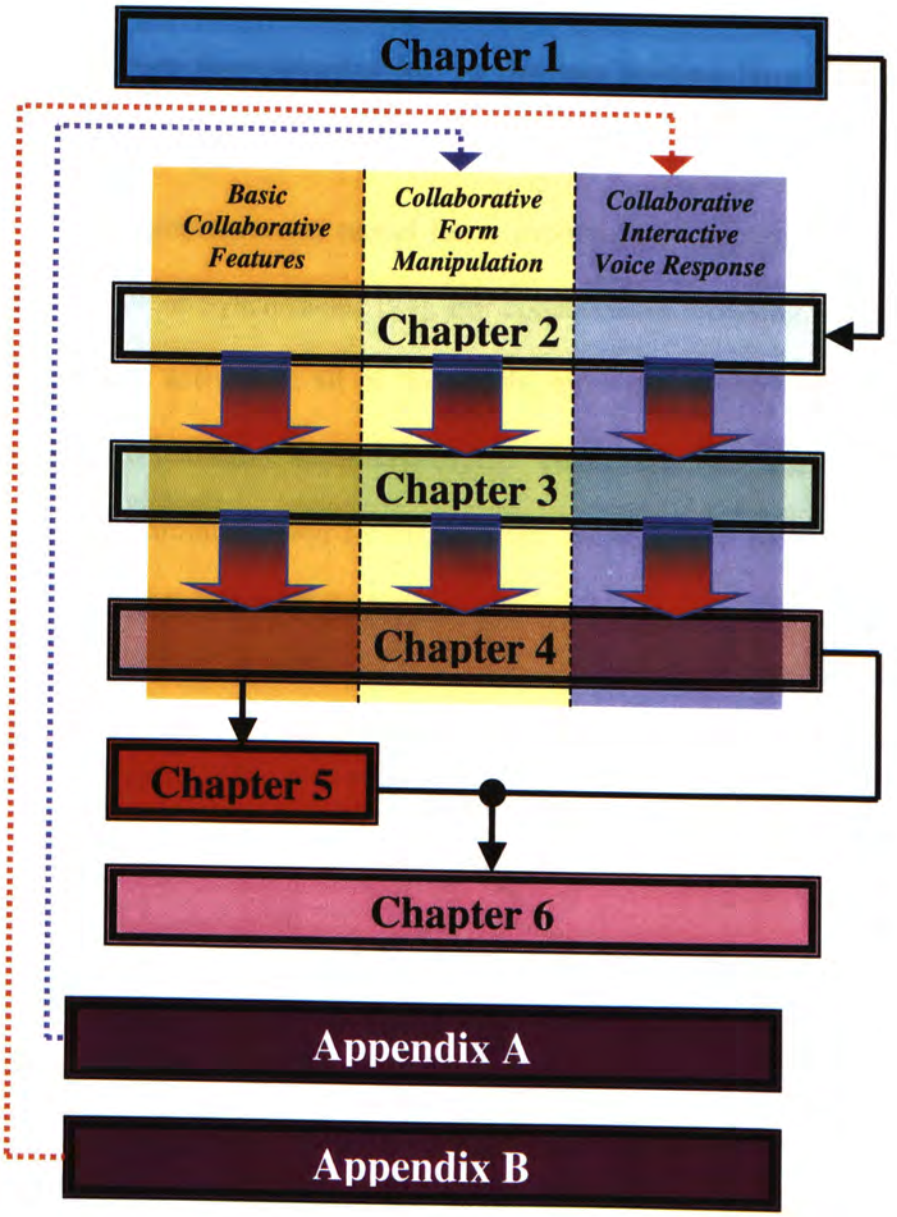


Figure 2 The Dependency of the Thesis Chapters

2. Features of Collaborative Browsing

2.1. Feature Synchronization vs Bitmap Screen Transfer

The communication model for collaborative computing can be classified as Asynchronous or Synchronous [21]. For collaborative browsing, we can regard it as a synchronous activity as all of the collaborative operations between customer and operator are real-time communications. There are two common methods for information communication in synchronous collaborative browsing. They are the Bitmap approach and the Feature approach.

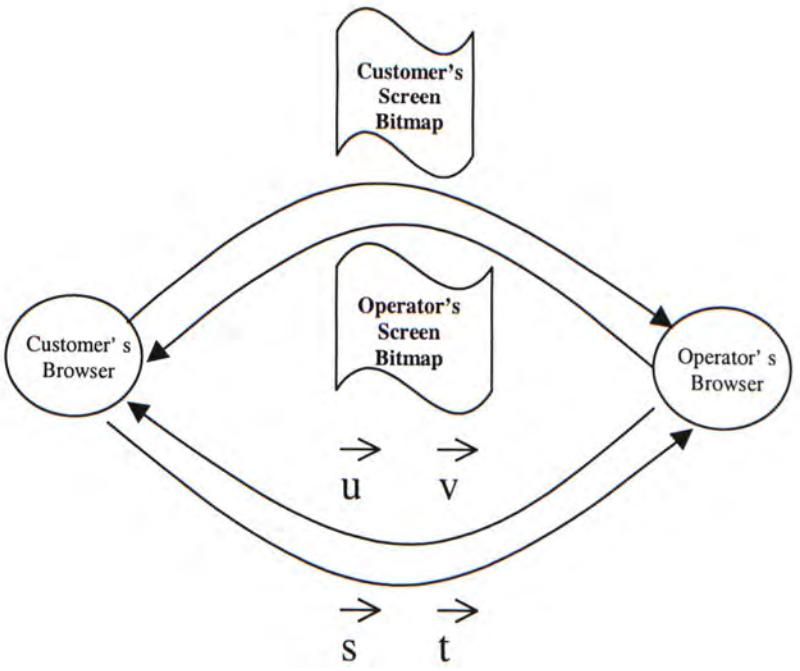


Figure 3 Feature Synchronization versus Bitmap Screen Transfer

The Bitmap approach refers to the synchronous screen transfer between the operator side and the customer side browsers. Any user input, such as mouse drag or key press and display output are also synchronized. The key advantage is the

absolute consistency of the synchronized contents between the two involved parties. Thus, it is application independent. However, there is a great bandwidth consumption for such a kind of synchronization as every screen change on one side triggers the transmission of the corresponding screen bitmap to the other side, no matter whether it is relevant to the synchronization focus or not. This approach is feasible only for bandwidth abundant environment, like a LAN.

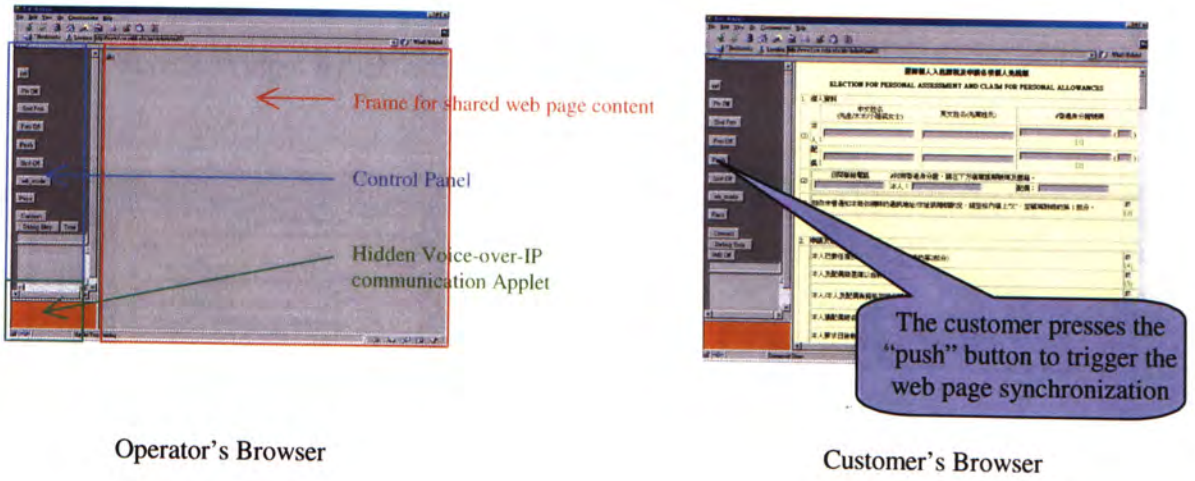
The type of communication that we have used for collaboration is the feature approach. In contrast to the bitmap approach, the feature approach focuses on the synchronization of the only relevant and critical information between the participating parties. In case of collaborative browsing, which deals with the synchronization of web contents, the critical features then include the URL address, browser scrollbar position, HTML form data, etc. The obvious advantage is the significant reduction in the bandwidth consumption during synchronization. However, the feature approach is application-specific in such a way that different collaborative applications require different sets of features for data synchronization.

2.2. Basic Collaborative Features of the Collaborative Browser

In general, a collaborative browser focuses on the content synchronization [22]. As soon as a customer and his serving operator have pressed the buttons respectively in their control panels, a collaborative browsing session begins. In a collaborative browsing session, the basic features supported in the collaborative browsers are:

2.2.1. Web Page Pushing

1)



2)

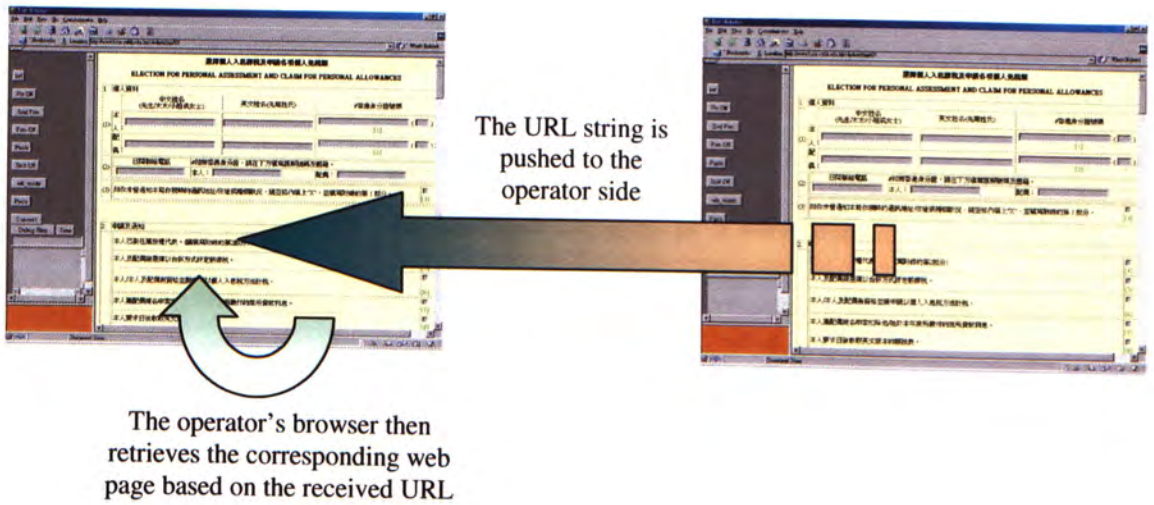


Figure 4 Web Page Pushing

When a session user shares the local display content with another remote user in the same session by pressing the "push" button, only the URL of the web page plus the browser scrollbar positions are transferred to the receiving end. After that, the receiving end fetches the web page based on the URL string and scrolls to the

corresponding scrollbar positions. This can be regarded as page feature pushing, instead of the bitmap image pushing. A lot of bandwidth is saved.

2.2.2. Screen Widget Synchronization

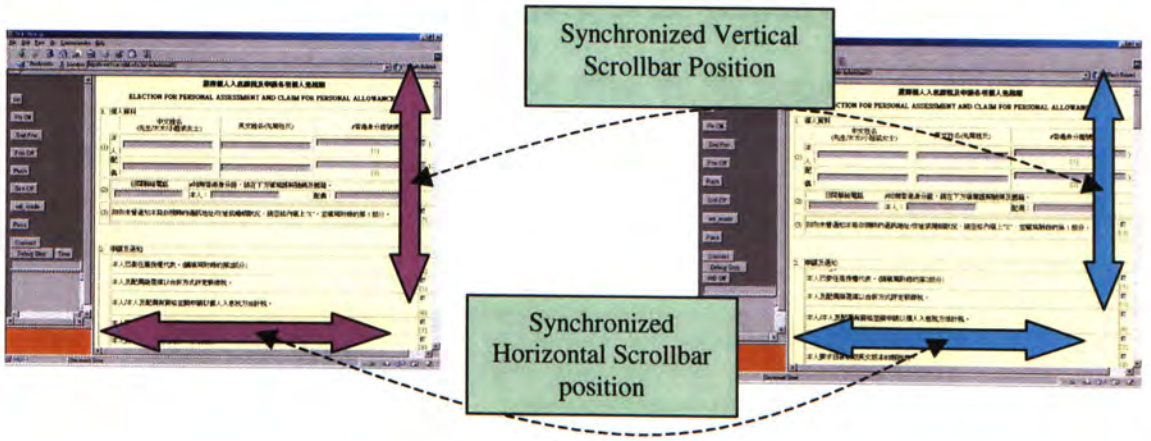


Figure 5 Widget Synchronization

Widgets refer to the user interface controls of a web browser. They include the horizontal and vertical scrollbars of the HTML content frame in the browser window. After synchronizing the web page contents, any change in the scrollbar (a widget) position on the pushing side will be automatically updated to the remote receiving side. Thus, the viewing scope is also synchronized in the browsers of the two session hosts. This is especially useful when the pushed web page is large in size.

During synchronization, the user of the receiving side is not allowed to change the widget settings unless he has gained the control, which will be discussed in a later chapter.

2.2.3. Tele-pointing and Shared Whiteboard

After synchronizing the same web page content between the session users, the user on the receiving side can track the mouse cursor controlled by the user on the remote pushing side by tele-pointing [23]. On the receiving side, the user will see a tele-pointer, which emulates the mouse cursor on the pushing side, on the local browser window.

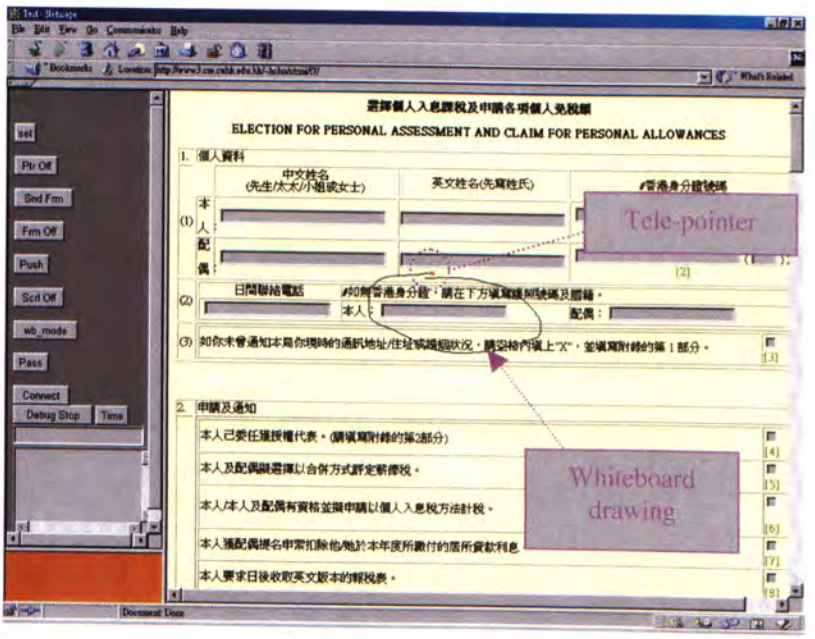


Figure 6 A Snapshot of the Tele-pointer and Whiteboard Drawing at the receiving side

The shared whiteboard mechanism can be regarded as putting a transparent HTML page onto the current HTML page. The session users can draw onto the transparency by simple mouse-clicks and mouse-drag operations. Then the drawings are updated in the collaborative browsers of both sides. However, it should be noted that no matter how the whiteboard page is drawn, the underlying HTML content page is not affected.

The synchronization mechanisms of the tele-pointing and the whiteboard data are very similar in the view of the transmission data types and the transmission methodology. Both services require the sequential transfer of a set of two-dimensional points.

In tele-pointing, the point coordinates are immediately transferred to the remote end as soon as they are sampled upon any mouse cursor movement on the pushing side. The transfer of the whiteboard data can be regarded as a repetitive version of the tele-pointing one. The details of the whiteboard implementation will be discussed in Chapter 4.

2.3. Collaborative Form Manipulation

2.3.1. Importance of Electronic Form Collaboration

A simple web page browsing may not require any particular collaborative support by the call center operator. When compared with other web page features, such as hyperlink, page text, web form possesses the highest user interaction complexity, such as filling in various form fields. Thus, the more complex a web form, the higher the user interaction complexity and the more likely the client seeks support from a call center operator. Thus, an operator-guided collaborative form filling support is necessary for collaborative browsing of web forms.

2.3.2. Basic Support for Form Collaboration — Data Synchronization

Sometimes, a web page contains an HTML form, through which a user can submit data through the Internet. Therefore, a collaborative browser needs to synchronize the data contained in a form as well as other web page contents within a collaborative browsing session. In form data synchronization, any data change in the form fields, such as checkboxes, radio buttons, text fields, in one side automatically invokes the data synchronization of the other side. As a result, the browsers of the customer and the operator share the same set of form data values in a common session.

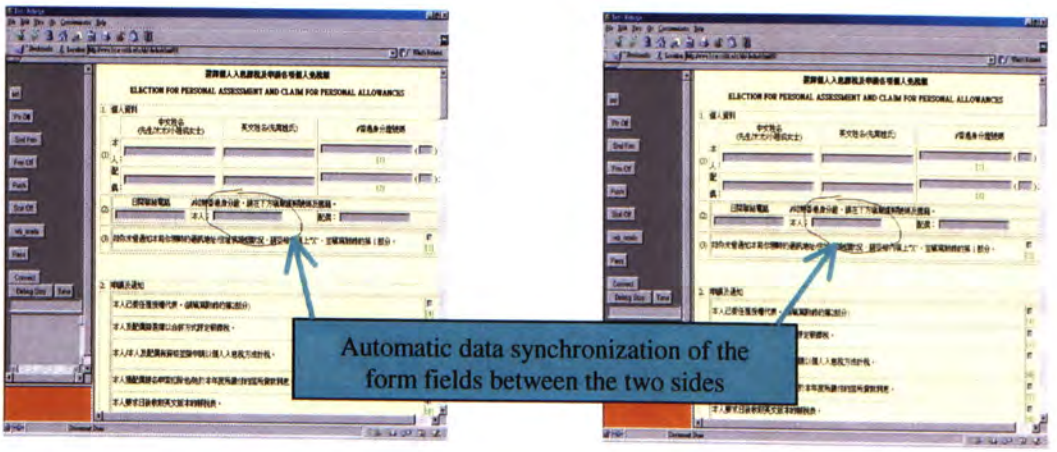


Figure 7 Form Data Synchronization

Tele-pointing service, shared whiteboard and the collaborative manipulation of form data are mutually independent of each other because all these collaborative services work on different layers even though they visually work on the same web page in a common session. No matter what a user has drawn in an HTML page form

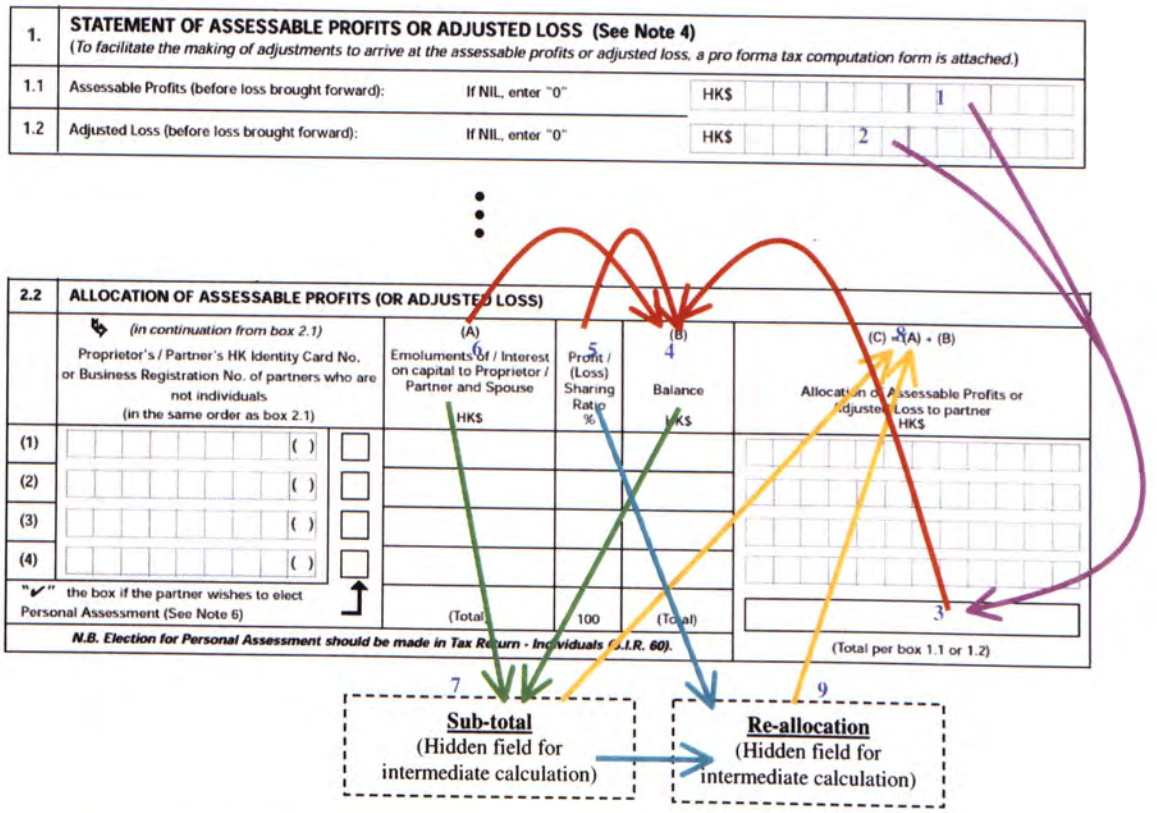
using a shared whiteboard, the data in each of the form fields are not altered. This layering architecture will be discussed in a later chapter when we discuss the Document Object Architecture (DOA).

2.3.3. Existence of Form Complexity — Form Data Dependency

In most documents, dependencies [24], [25] among the data exist. Simply take this thesis as an example. There are some relationships, or content dependencies, among all the thesis chapters as depicted in the diagram of section 1.3. Visualization of such a dependency graph assists a reader in traversing the thesis content more systematically. Similarly, a web form, as a kind of web document, carries data dependencies [26] that are useful to a form user. Understanding of such dependencies may often help the user to handle the forms better.

When an HTML form consists of many data fields, a user needs to input an enormous amount of data values for different fields. The form complexity is related to the difficulty of completing the form. Inside a form, some form fields, which appear to be independent of each other, actually store closely related values. If facilities are provided to the operators to interpret the relationships of the form fields, they can properly handle the customer's requests better.

For example, the following diagram is an extracted part of a real profit tax return form in Hong Kong (See Appendix A for the details). The fields labelled with a number carry some data dependencies that are represented by arrows. The details of the data dependencies are described in the following box.



Steps to fill in this part of the form:

1. (3) = (1) – (2) (i.e. pink arrows)
2. (4) = (3) – (6), using (5) as the weighting (i.e. orange arrows)
3. (7) = (6) – (4) (i.e. green arrows)
4. Adjust the entries in (7) using (5) the weighting so that each entry in (7) is either all positive or all negative, then save the adjusted differences into (9) (i.e. blue arrows)
5. (8) = (9) – (7) (i.e. yellow arrows)

Note:

- An arrow represents a data dependency link. For example, the arrows from (1) to (3) and from (2) to (3) mean the value in field (3) depends on the values in field (1) and field (2).
- Please refer to the appendix for the details of this example.

Figure 8 Data Relationships in a Profit Tax Return Form

From the previous example, filling in a form is not as easy as inputting sequential data to each field. Rather, understanding the data relationships is often necessary.

Moreover, the form complexity becomes higher when the number of form fields increases, bringing with them even more complicated data relationships. Unless the user understands the data relationships among the fields, he will face tremendous difficulty to fill in the form fields.

2.3.4. Hinting

When requesting input information, current HTML forms will display a set of user interface prompts, such as text fields, checkboxes, without any logical data relationship being shown.

However, in a collaborative browsing session, simple synchronization of the browsers of the user and the operator seems inadequate for collaboration because the forms themselves do not show the complicated data relationships hidden among the form fields. To enhance the form data collaboration, we need a way to identify the data relationships among these fields.

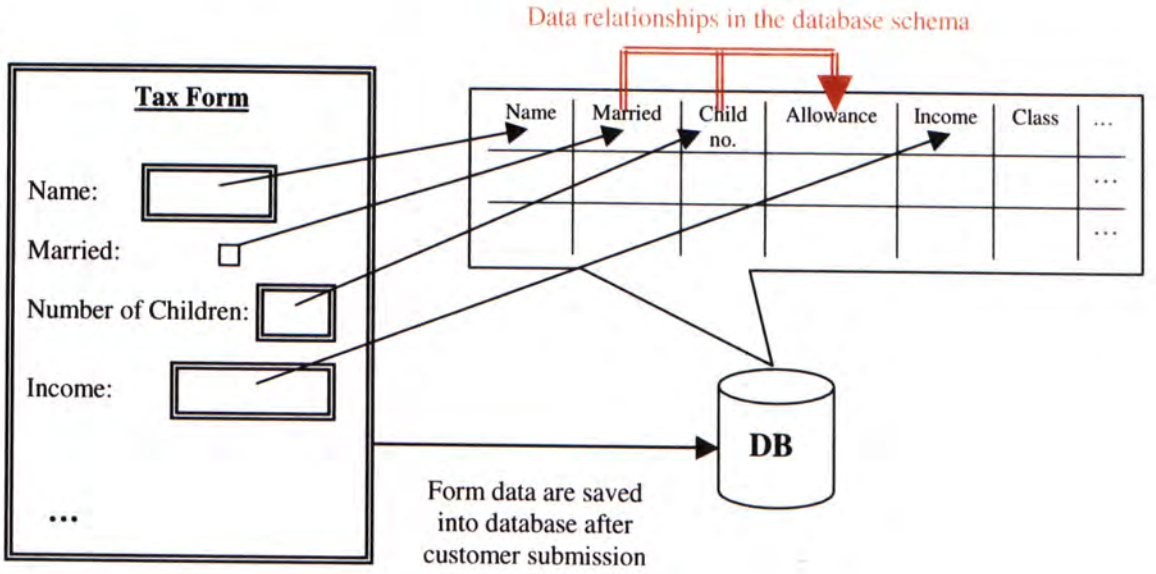


Figure 9 Associating a form with a database

In some cases, the form data entered in web browsers will finally be submitted to a relational database, which serves as a data repository of the form field data. A relational database, which is well structured, contains some meta-data and pre-defined field relationships, such as data dependency or data integrity rules. Thus, linking the form data fields with the database can be useful for visualizing the hidden data relationships.

For example, Figure 9 shows a tax form and the associated database table at the backend database. Normally, the data entered in the tax form are most likely to be stored into a database for internal reference by the government. In other words, each form field somehow corresponds to one or more identical table fields in the database. Therefore, we may regard the form fields as a web-based representation of a corresponding database schema. Yet, from the view of a user, who has to fill in the

form, the form fields remain to appear to be independent of each other during the form filling process.

However, one may notice some form field relationships that are not visually shown in the form. Suppose there is a rule that if a lady is divorced and has at least one child, the government will give her a larger personal allowance to relieve her tax burden. But this lady encounters difficulty in comprehending the rule while she is filling in the tax form. So, she calls an operator for assistance.

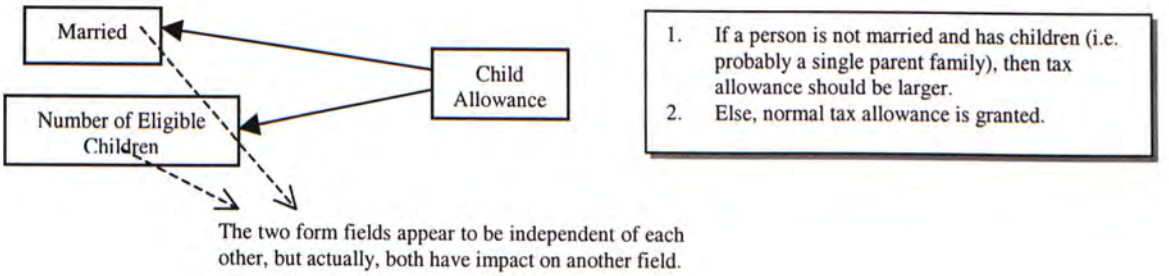


Figure 10 Graphical representation of the data dependencies of an example tax form

They may have the following dialog:

Lady : I have learnt about a tax allowance problem.
Operator : Yes. Please tell me your problem.
Lady : I have problem of filling in a form field 'child allowance' as I am divorced.

At this point, the operator asks the lady for permission to share her web form through the Internet. Looking at the form and database tables using a searching methodology to be discussed, the operator locates the form field 'child allowance' and its dependency on two other form fields values: marital status and number of children. She also learnt that the lady has two children. The dialog continues:

Operator : Madam, how old are your two children?
Lady : One is 22, the other is 15.
Operator : Are you supporting the second child?
Lady : Yes. My former husband never gives us any financial assistance.
Operator : The rule states that you can deduct $10, 000 for each child under 18. In your case, your second child is eligible for such allowance.
Lady : Thank you.

The above data dependencies are called Hints. In general, Hints refer to the hidden data relationships that are not shown in the form directly but useful for guidance when filling in the form. Usually, Hints are specific to different scenarios. In the case of a tax form, Hints refer to the internal table field relationships induced from the set of complicated tax rules or calculations not directly shown in the tax form. In the above example, the dependency of child allowance on marital status and number of children is an example.

Hinting refers to the use of specific hints by an operator to handle form data during a collaborative session with a user. Figure 11 shows the framework of the hinting support system.

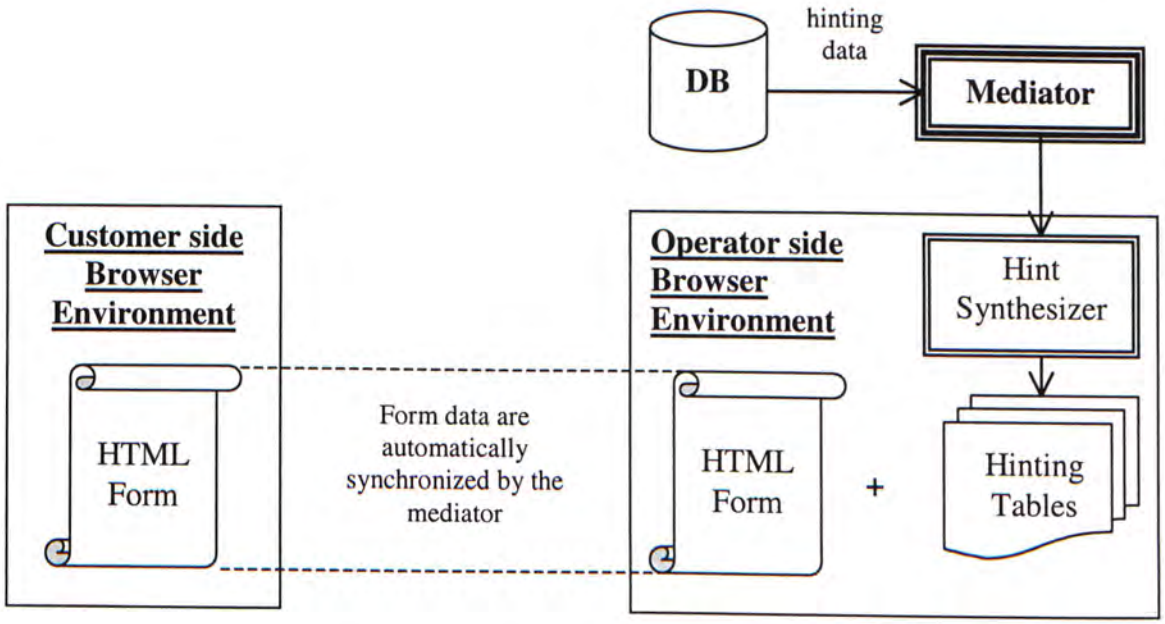


Figure 11 A Hinting Support System

Hints are presented to the operator in form of simulated database tables, called Hinting Tables. During collaboration, using a form field, like child allowance, as an index, form-specific Hints are retrieved from a backend database to the

operator side browser. Then a set of Hinting Tables, which have been previously synthesized, are displayed on the operator side. The Hinting Tables can be regarded as the simulated view of the database tables associated with the HTML form. They help the operator to know how various form field data are represented and transformed into the database table fields and the mutual data dependencies among the table fields at the backend.

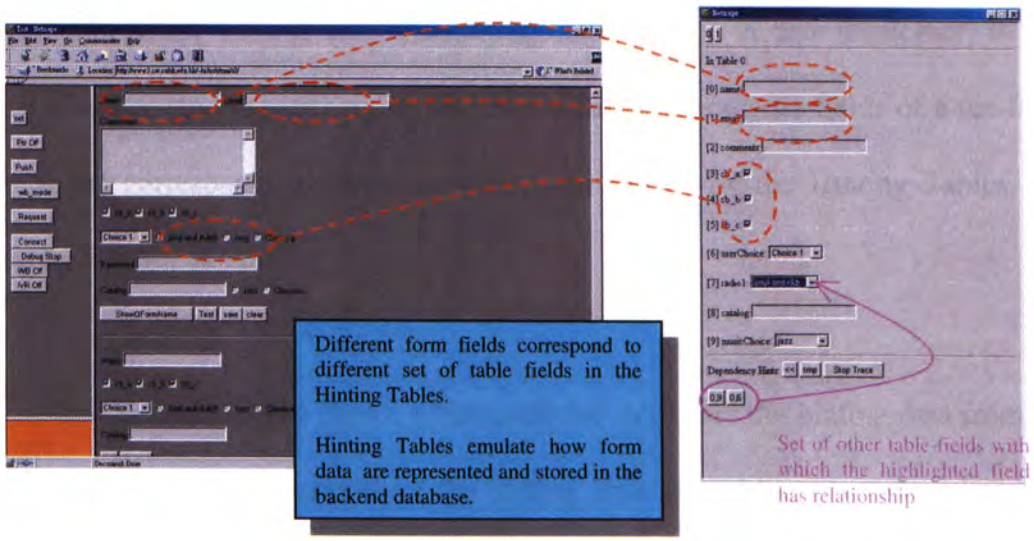


Figure 12 A Form and the corresponding Hinting Tables

With Hinting, the form fields become more meaningful to the operator as the operator can identify the underlying relationships among the form fields by tracing the data dependency among the table fields in the Hinting Tables. It should be pointed out that supplementary notes can be appended to the table fields to help the operator to play a more active role in the dialog with the user. In the child allowance example, information for the definition of an eligible child is certainly very important. Yet, the definition, which is not appropriate to be included in the form,

can appear as appendix information associated with the table field: number of children.

Hinting Tables are presented to operators but not directly to customers because customers are not familiar with the representation of the form data in the backend database tables while operators, who are well-trained staff, should have understood the data relationships among the form fields and the corresponding tables fields before. Hints of the Hinting Tables serve as additional reminders for the operators to locate various data relationships more easily. A user (customer) is less likely to understand the underlying data relationships among the fields of a tax form even if the corresponding database tables, represented by the Hinting Tables, are given for his self-reference.

It is unnecessary to derive a hint extractor to extract the hinting data from the database because the generation of the hinting data should have been conducted at the design time of a form and the associated tables. During the schema design [27] of the tables, the data dependencies (the hinting data) among the table fields and data integrity rules must be specified. That means the hinting data should have already been manually specified with respect to the application context and stored into the database at design time before the form is generated. For example, the hinting data for a tax form should have been specified according to the set of tax rules and calculations during the design of the tables.

The schema design [28] has significant impact on the topology of the dependency relationships. Moreover, the mapping from the form fields to the table

fields and vice versa, are not necessarily one-to-one since one piece of form data may serve multiple data processing functions and one data processing function can involve many data from the form fields.

For those forms that require no database for form data repository, these forms, like a simple account query form, are mostly so simple that no hint is necessary.

2.4. Collaborative IVR

2.4.1. Traditional Interactive Voice Response (IVR) Service

Interactive Voice Response Service refers to the automated customer service provided by the touch-tone dialing based user interaction. When a phone user dials to a call center system, he is prompted to select a list of items by pressing the buttons in the phone keypad. After several selections, he can finally acquire a particular application service, such as information query. The entire interaction methodology [29] of a typical IVR phone service is based on the phone-based user input and the corresponding voice-based system response.

An example is phone banking. Suppose a customer has dialed to a call center for IVR service:

System	**: Welcome to the Phone Banking IVR Service, Please press '1' for Account Manipulation, '2' for Financial Services.**
Customer	**: [Press 2]**
System	**: Please press '1' for Balance Query, '2' for Fund Transfer, '3' for Deposit Renewal.**
Customer	**: [Press 2]**
System	**: Please enter the account number from which the fund is transferred.**
Customer	**: [Press 123456789]**
System	**: Please enter the account number to which the fund is transferred.**
...	

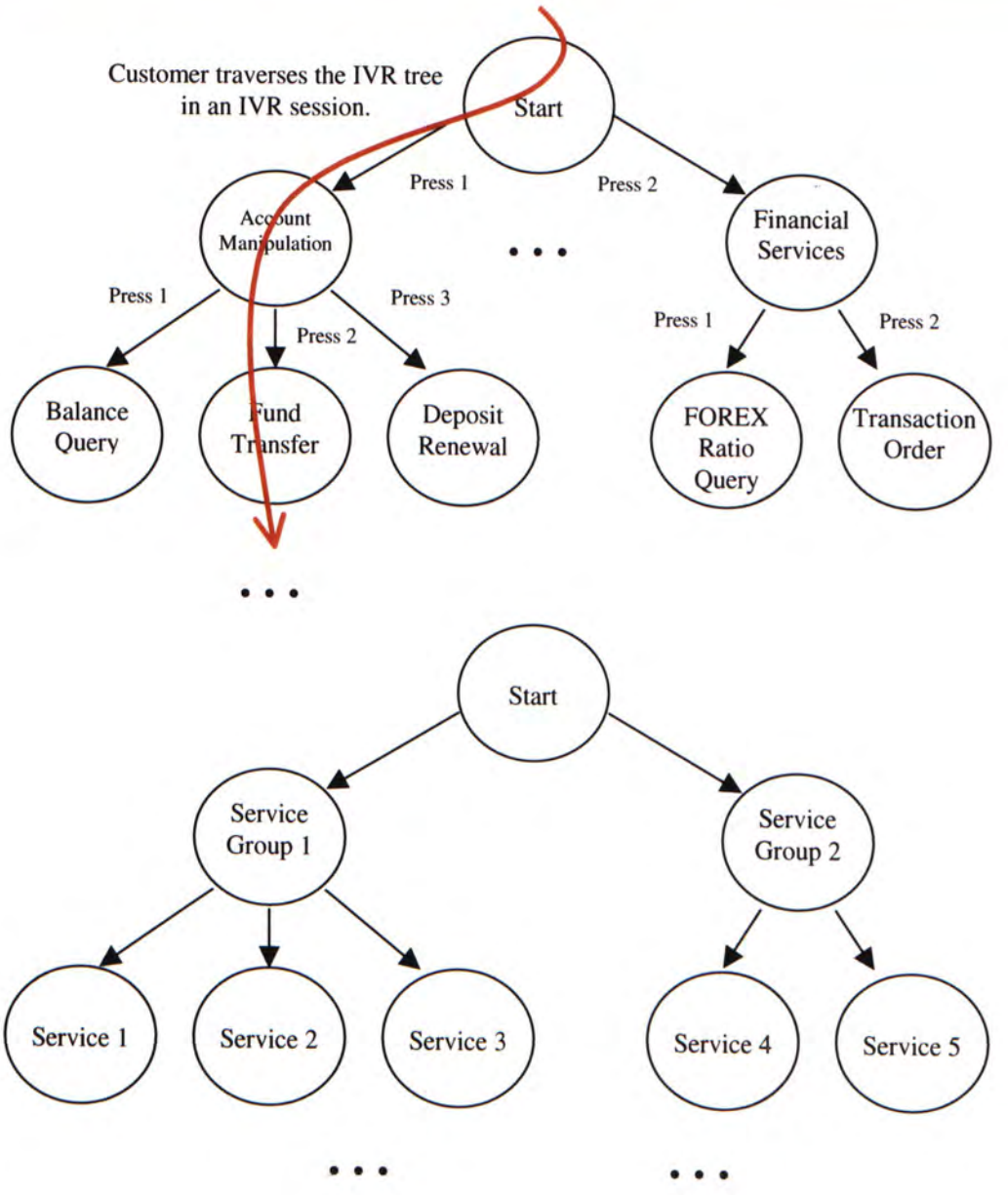


Figure 13 Conventional IVR service architecture

Conventional phone-based IVR service architecture can be graphically represented as a tree of service nodes [30]. Figure 13 is the schematic representation of a phone-based IVR service. Each edge traversal is realized by an IVR operation (pressing a touch tone key of a phone).

However, most current IVR services are limited by the highly restrictive user interface of the phone keypad on the customer side. For example, if there are many services or functions deployed in a phone banking IVR system, the resulting IVR tree can become very tall and fat with many terminal service nodes. Users will have to key in the phone buttons many times in order to acquire a particular service. It is possible that users can easily forget what they have input and they can 'get lost' in the complex IVR tree. As a result, a traditional IVR tree cannot be very complex in terms of the tree levels and the number of tree nodes. However, simple IVR trees limit the amount and the functionality of the services that can be provided by their corresponding IVR system.

2.4.2. Abstraction — Correlating Form Interaction Mechansim with IVR

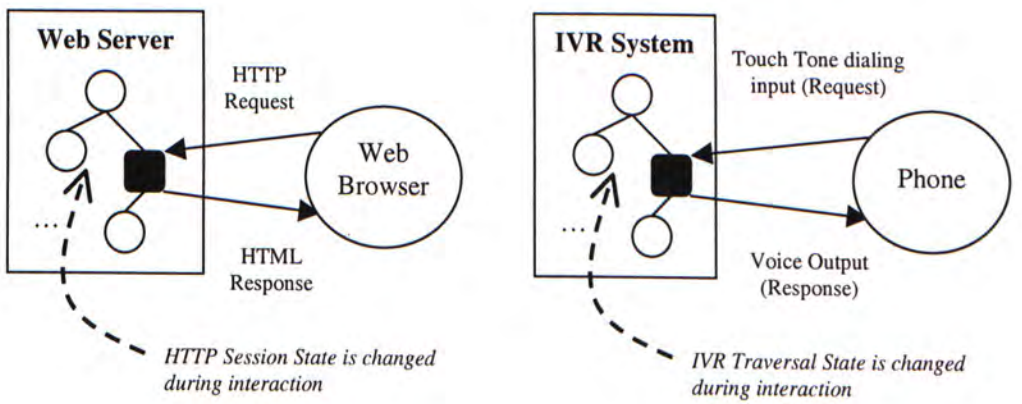


Figure 14 Similarity between Form Interaction and IVR

The HTML form interaction mechanism [31],[32] is based on a request-response methodology for browser-server communication. A web browser issues an HTML form data submission request to the remote web server while the web server responds this request by sending an HTML file containing the submission acknowledgement details back to the browser.

Interestingly, IVR service is also based on the same interaction methodology. For example, a user inputs a phone key to generate the request to the IVR system. Then, the system may access to a database with respect to the request if necessary, and automatically responds through the voice output that is normally synthesized by playing the pre-recorded audio files or a text-to-speech engine, to the user.

The similarity [33] of the interaction mechanism between the HTML form and the IVR system makes it possible to re-use the form interaction mechanism for enhancing the traditional IVR systems.

2.4.3. Collaborative IVR by Form Interaction Mechanism

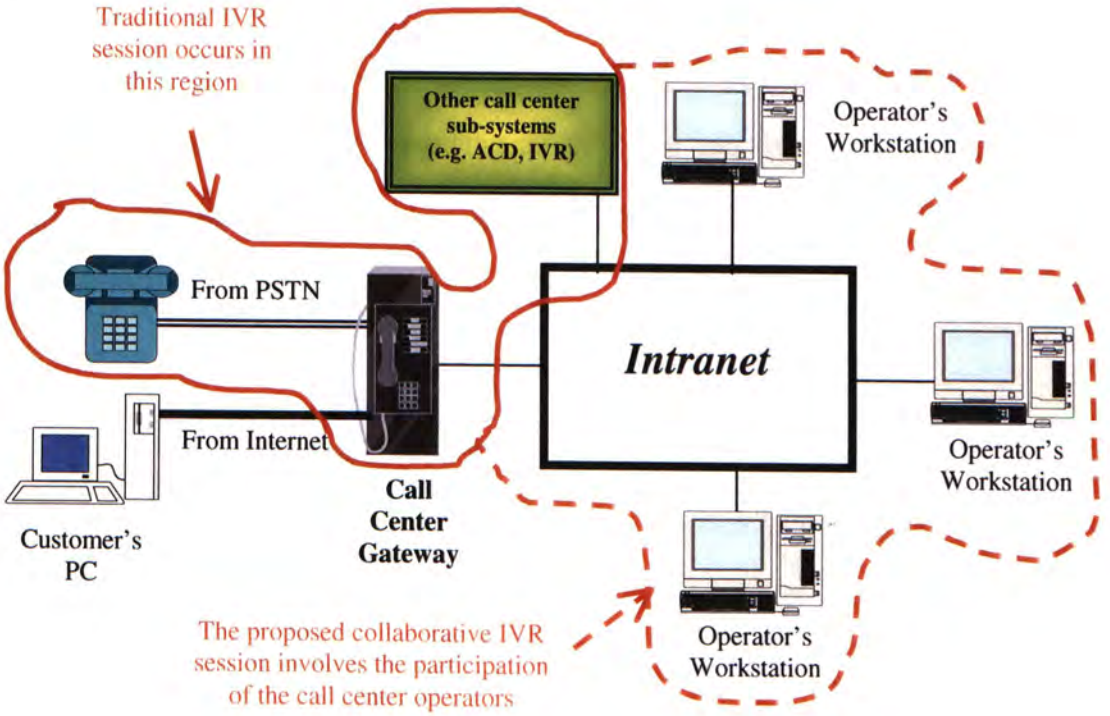


Figure 15 Extending IVR to operator-assisted collaborative IVR

Figure 15 shows that the traditional IVR service involves only the call center gateway for telephone call connection and the IVR sub-system for IVR phone service.

Knowing that both HTML form and IVR service are based on request-response interaction methodology, it is possible to use this form interaction mechanism as the communication medium between the customer and the operator. Our goal is to revolutionize the user interaction methodology of the traditional IVR system used in call centers. After associating the current IVR interaction methodology with the HTML form interaction one, collaborative form interaction mechanism that has been established can be re-used to support collaborative features in traditional IVR systems [34].

There are some similarities between collaborative browsing and collaborative IVR. Traversing an IVR tree is analogous to web surfing. In the collaborative browsing system, a customer can contact an operator for support service during web surfing. Similarly, a customer traversing an IVR tree can dynamically contact an operator for IVR-specific support service through the phone, instead of a browser.

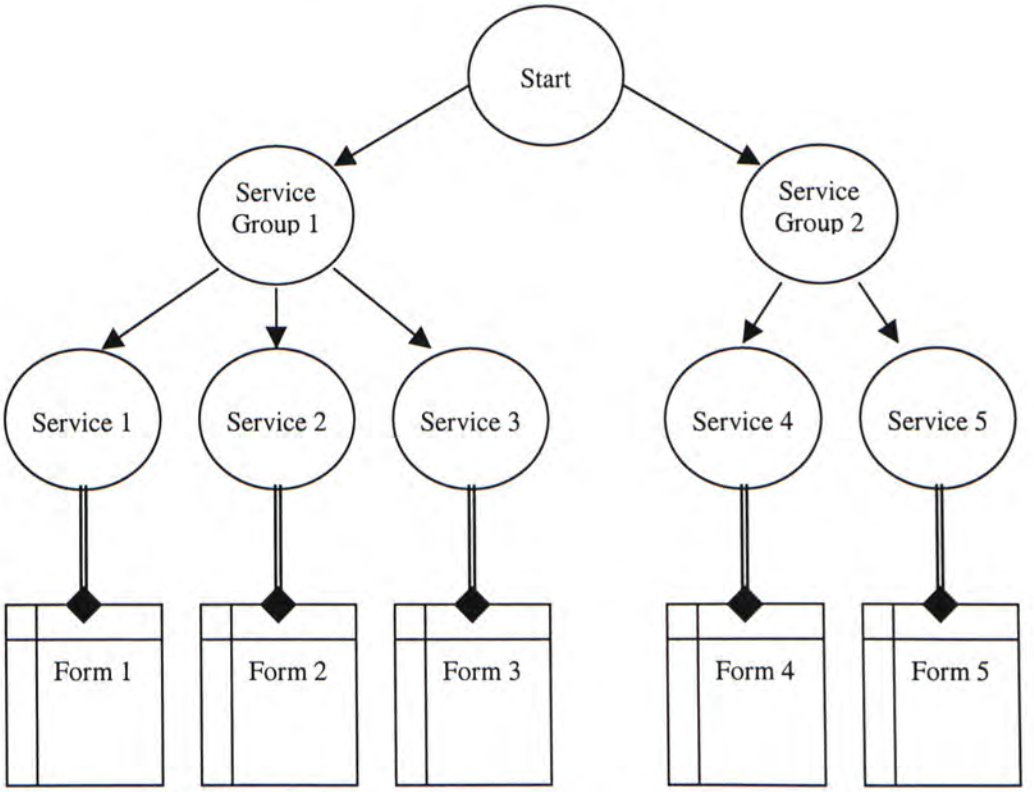


Figure 16 The Modified IVR service tree

In collaborative IVR, a customer firstly traverses the IVR tree to access the target service just as in a traditional IVR system. When he has a problem on accessing a particular IVR service, he can press a phone key to contact the remote operator who can gives guidance related to the IVR service. Once, the terminal service node (the target service) is reached, the user is prompted to input some IVR service data through his phone. Then, the IVR service data captured from the

customer are transformed into a set of form data inhabited in a special HTML form, called Telephone Form. When the customer has problems on entering the IVR service data, he can again contact the remote operator who can perceive the IVR data entered by the customer through the Telephone Form displayed on a screen. The operator can help the customer fill in the IVR data by manipulating the data in the Telephone Form.

The critical advantage is that the call center operator can simply use a browser to view the IVR data, captured from the customer and represented by a Telephone Form, for IVR data collaboration. Also, the call center operator can interact with the customer by simply modifying the form data while the communication method at the customer side remains unchanged (i.e. using a simple telephone). Thus, the internal Telephone Form mechanism for customer support is totally transparent to the customer.

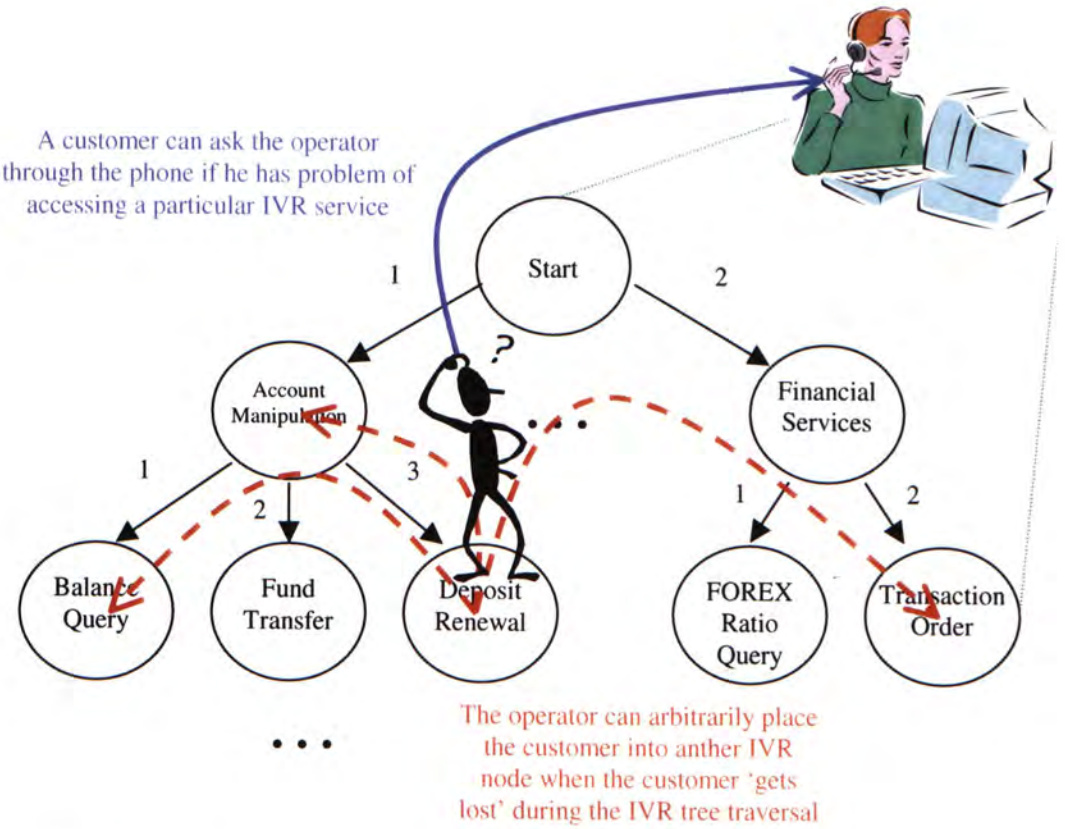


Figure 17 Customer and operator collaboratively work on the same IVR service tree

Using foreign exchange rate query in phone banking (See Appendix B) as an example, a customer wants to query the exchange ratio of US Dollar to Japanese Yen. He then presses '2' followed by '1' to reach the terminal service node for foreign exchange query. The customer is now required to input the representation codes of the two currencies through the phone keypad. From the view of the system backend, the customer is now required to fill in the Telephone Form for this particular bank service. After that, a query result will be given to the customer through the voice output by the system.

It should be noted that the customer does not know whether he is currently filling in a Telephone Form, as the customer is only prompted to enter some numeric

values sequentially from the phone keypad in a request-response manner. There is no change in the user interaction pattern. The underlying Telephone Form mechanism is highly encapsulated to the customer but only used by the operator at the backend.

The foreign exchange example is very simple. In real life, a customer may 'get lost' in a complicated IVR tree with many service nodes due to input mistakes or wrong interpretation missing parts of the voice prompts. Then, he can simply press a particular phone key to seek the support from the call center operator. For example, if a customer wants to query a foreign exchange rate but he is now in the 'Balance Query' node, he can simply press a particular hot key and ask the operator for the key sequence to travel to the IVR node for foreign exchange rate query. Alternatively, the operator can even assign the customer to the target service node directly using the operator's browser with the use of IVR Hinting, which will be discussed in a later chapter.

As a whole, we have described two types of call center services. One is operator-assisted customer support, where human operator interacts with the customer for customer support. The other is the self-served support service, where the customer acquires the support service on his own. Traditional IVR service belongs to the latter one. However, under the proposed Collaborative IVR environment, a customer can interact with the IVR system of a call center with the collaborative assistance of a remote operator. As a result, the traditional self-served IVR service is transformed into an operator-assisted collaborative service for enhanced customer support. This makes the entire IVR service interaction more user-friendly.

Moreover, as described earlier, a traditional IVR system can only provide a set of simple, unattended services due to its simple IVR trees. With the support of operator side collaboration, it is feasible to have a more complicated IVR tree providing more diversified IVR customer services. This is because supporting collaboration feature in the IVR services relieves the IVR operational difficulties faced by the customer and increases the flexibility of IVR service customization. As a result, we can have a highly customized IVR service tree for different customers to provide personalized services because there always exist call center operators to help the customers who 'get stuck' in a particular IVR tree node.

3. Software Architecture

3.1. The Three-Tier Architecture

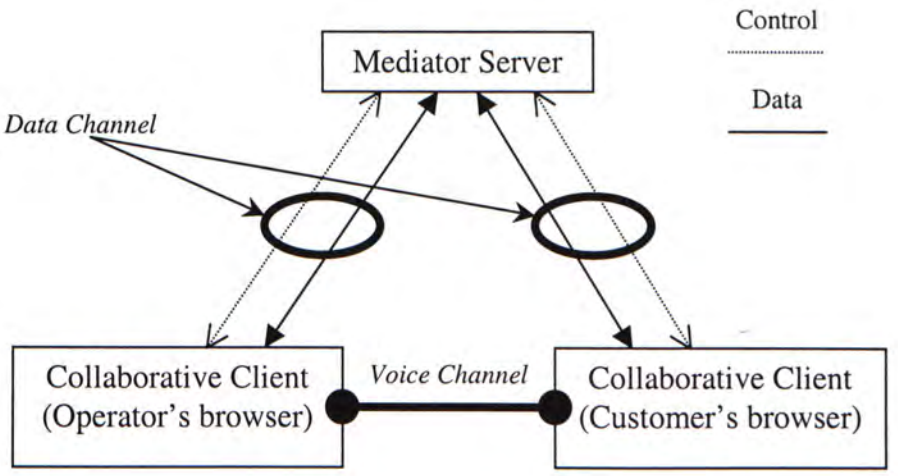
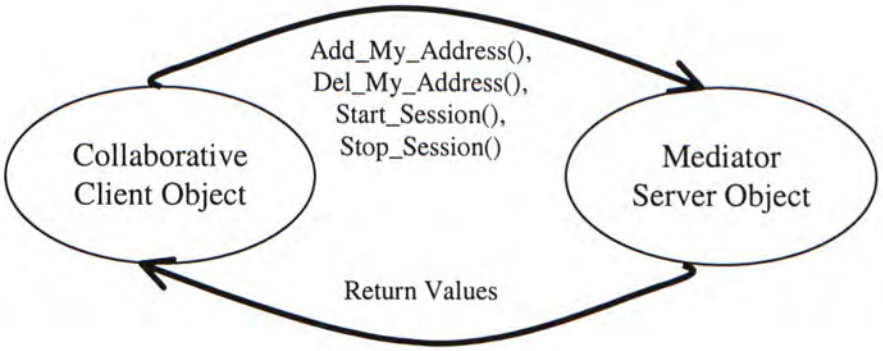


Figure 18 The Three-tier Architecture

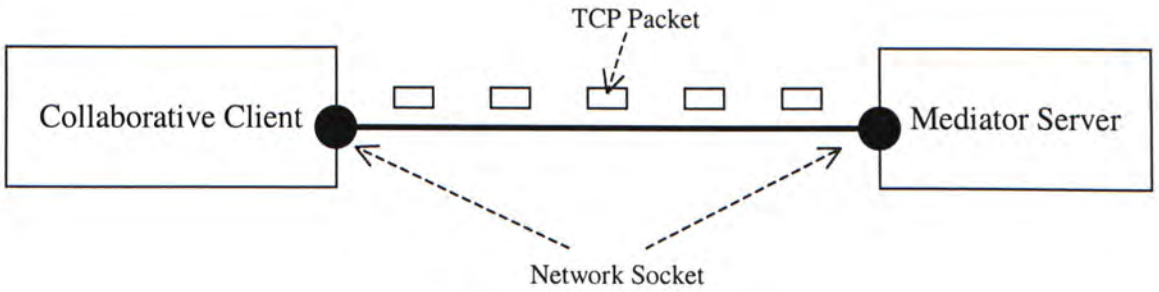
In the three-tier model, the mediator server can run in either the gateway computer or a separate machine in the Intranet. The operator's browser is in one of the workstations in the call center system while the customer's computer can be any host from the outside Internet.

There are totally two communication channels as depicted in the above figure. One is for the collaborative browsing. This includes the transfer of whiteboard data, tele-pointer geometry and form data. The other is for the control signalling, which includes the initialization and termination of collaborative sessions and addressing the customers' computers. Transmission Control Protocol (TCP) [35]

can be used for the data communication channel while Remote Method Invocation (RMI) [36] of distributed object orientation can be used to deliver the control signals.



A) Remote method call in the control channel



B) TCP packet communication in the data channel

Figure 19 Different Communication Methodology in the Control and Data Channels

In general, there are two alternatives for the transport layer communication methodologies [37]: User Datagram Protocol (UDP), which generates lower overhead but does not guarantee data delivery and TCP, which generates higher overhead but guarantees data delivery. Though TCP possesses overhead in the connection setup and connection maintenance, such as packet sequencing, it is used as the transport protocol for the collaborative browsing data because data reliability

is more important than real-time delivery performance, which is stressed in the voice channel. We need to guarantee reliable data transport in order to let the remote end-point to receive the right information in a correct sequence. Here, the packet acknowledgement mechanism of TCP is chosen.

RMI is the desired way for control signalling because this simplifies the communication architecture by replacing the low level socket networking with the high level method call of remote objects in the server side. Knowing that the control signals can only be regarded as a set of simple commands to instruct the mediator for session control operations in a collaborative browsing session, the real-time streaming criterion and the data delivery guarantee are not the considerations. We need not care about whether TCP or UDP is suitable, but only to send a single command correctly to the mediator upon client side request. Therefore, we can abstract the underlying networking techniques by using the high level remote object method calls, where the underlying networking matters are automatically handled by the corresponding distributed object middle-ware [38].

The mediator server is the middle tier of the whole architecture. It plays the role of data redirection between the two collaborative clients. Due to the network security of the Java Applet, the two collaborative browsers are not allowed to connect directly unless such a constraint is overridden arbitrarily [39]. Also, the mediator is responsible for internal data processing for other services, such as, hinting in collaborative form manipulation and the collaborative IVR.

In the collaborative browser, Java Applet is used for networking services, JavaScript is used for browser display rendering and browser plug-in [40] is responsible for whiteboard services. Figure 20 shows a typical collaborative browser user interface:

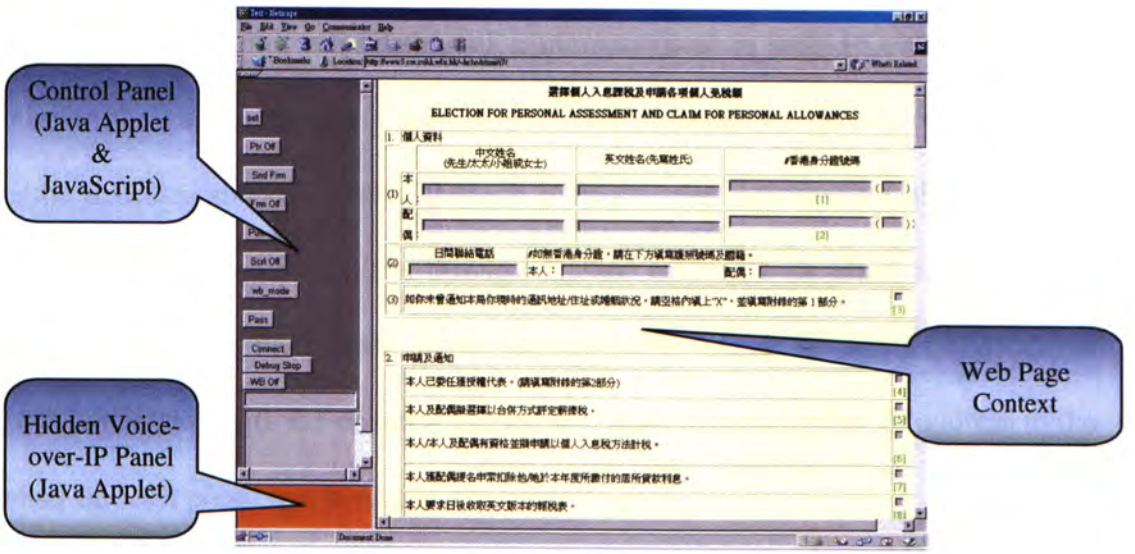


Figure 20 A Snapshot of the Collaborative Browser

The Control Panel is composed of a Java Applet and JavaScript objects. This is the core interface for controlling a collaborative browsing session. The Voice-over-IP Panel is composed of another Java Applet, responsible for the voice communication in the Internet, assuming PC-to-PC VoIP is available. The Context is the HTML frame that refers to the web page content shared between a customer and an operator. This is the place where the actual collaborative browsing activities take place.

3.2. The Collaboration Mechanism for Collaborative Browser

3.2.1. Session Initialization/Termination

The addressing information of a user/customer is easy to obtain if he/she accesses the call center using a PC-to-PC VoIP connection. On the other hand, if he/she accesses the call center through a PSTN, the Automatic Call Distributor (ACD) would identify him/her through his/her caller ID or other user/customer information. In the PSTN case, if the user/customer intends to start a collaborative browsing session with the operator, he/she will have to send his/her IP address to the center through pressing a button on the control panel.

Suppose that a user and an operator would like to invoke a collaborative browsing session during a voice connection, they need to trigger the connection by pressing the "connect" button in the control panel of the collaborative browsing. The collaborative browsing applets of their browsers will then make requests to the mediator respectively. The mediator's task is to ensure that both IP addresses eventually arrive. If so, a pair of TCP connections is made between the mediator and the two computers respectively. The mediator also spawns a new server thread that specifically maintains the connections between two computers. Otherwise, an exception signal is sent to both user and operator.

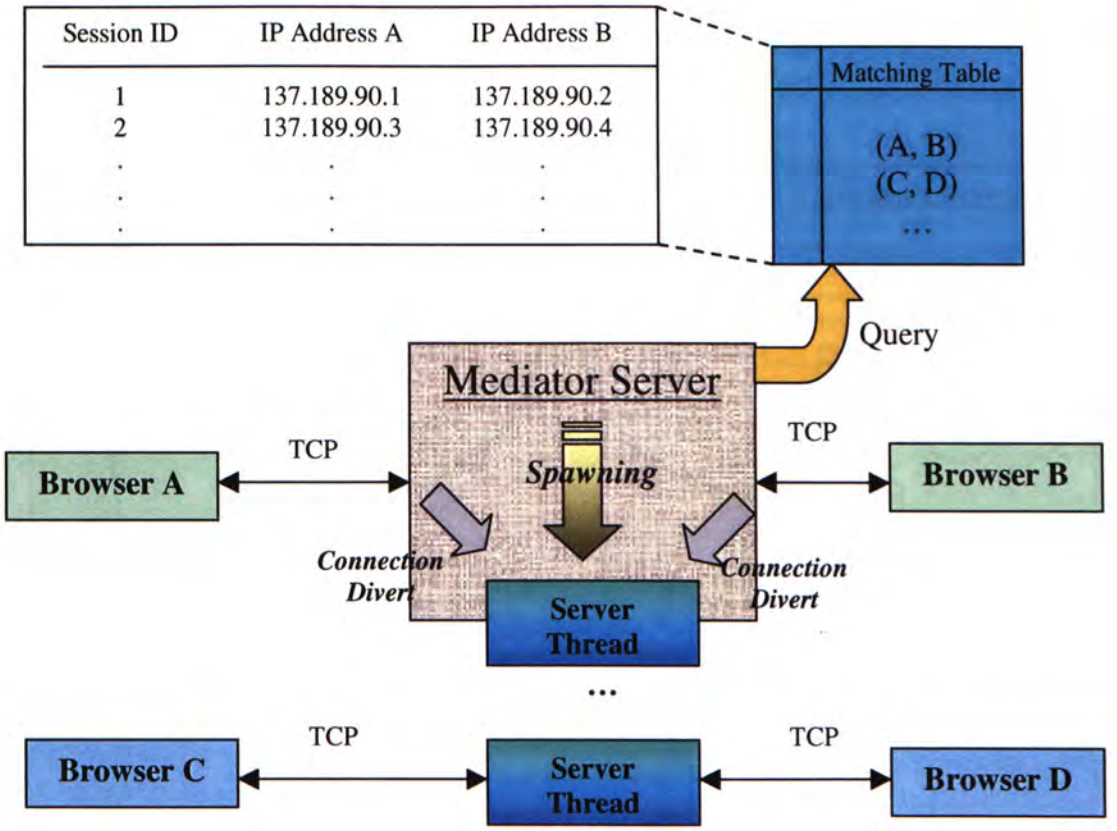


Figure 21 The Server Thread Generation upon Session Initialization

The mediator keeps the session information, which consists of the host IP addresses and a session ID, in the matching table for session management and maintenance. This ensures the data in a particular session are correctly routed to the corresponding pair of session users. Thus, multiple collaborative sessions can be supported at the same time.

Either the user or the operator can stop the collaborative session by pressing the "stop" button in the control panel. Then, the corresponding server thread will die and the pair of the TCP connections will be destroyed. However, the two IP addresses of the operator and the user will remain in the matching table until their voice connection terminates.

3.2.2. Data Flow of the Basic Collaboration Features

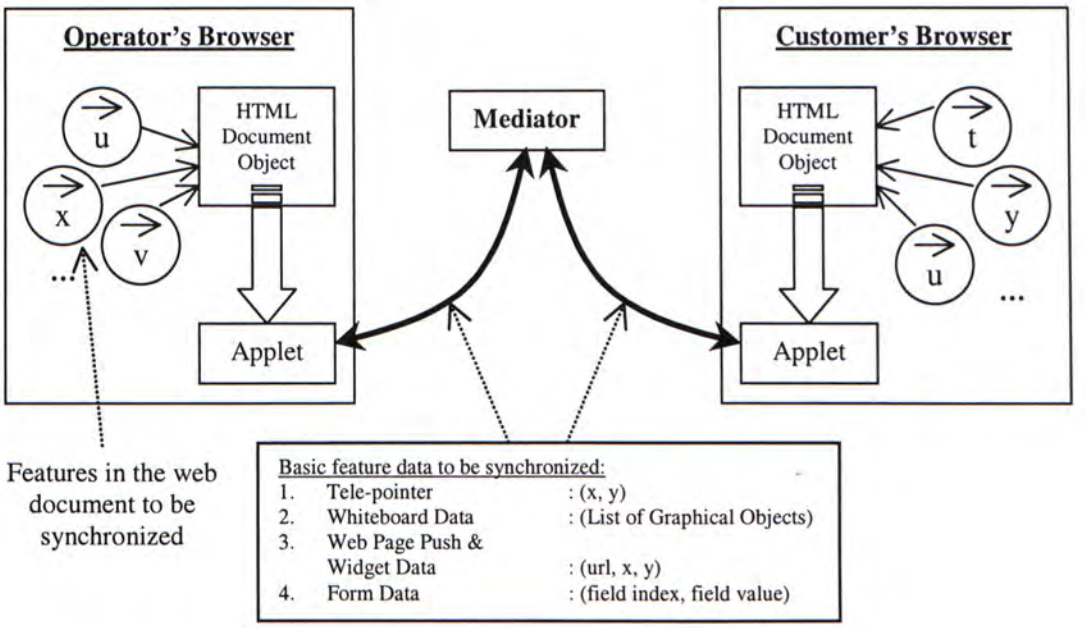


Figure 22 Feature Synchronization between a pair of browsers

In a particular collaborative browsing session, the entire synchronization methodology can be regarded as the synchronization of a set of feature data, like the URL of the shared web page, the geometric position of the tele-pointer,..., between the pair of collaborative browsers.

As shown in Figure 22, the HTML Document Object manipulates all the web document contents, such as form field data in a browser. It serves as the feature extractor that extracts the basic feature data from the web document currently shared by the operator and the customer, and directs them to the Applet for transmitting to the other side through the mediator. It monitors all changes in the features and notifies the remote side browser of the changes all the time during collaboration.

The role of the Applet is to transmit the feature data captured by the HTML Document Object to the Applet of the remote side browser.

3.2.3. Control Mechanism

A control and flow conflict problem will occur if we do not have arbitration management. Suppose that two session users draw on a shared whiteboard simultaneously. If there is no user control arbitration, one session user drawing on the upper part of a long and large HTML page may not notice the remote user is also drawing but on the far end lower part of the page. Then the two whiteboard frames on each side may display messy drawings. Rather, we prefer that one side can only view the whiteboard while the other side is drawing on it.

Thus, it is necessary to restrict that only one of the two session users can push data or control the entire collaboration at a time. A preemptive master–to-slave architecture is designed for such need. Call center operators are defined to be the master users while customers are the slave users. Upon session startup, the slave is given the control. The slave has the right to get the control without any permission from the master while the master can preempt the slave to get back the control at any time. However, if both the master and the slave race for the control, the master always wins because the master can hold the control permanently unless he releases it. In other words, the master has higher priority for control privilege than the slave. Still, the slave has the flexibility to actively gain the control if the master does not preempt or hold the control.

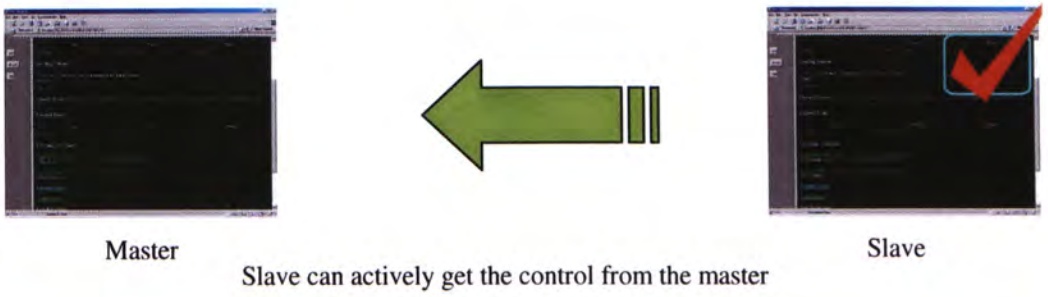


Figure 23 Master-to-Slave Control Mechanism

One may argue the slave (customer) will suffer from starvation if the master (operator) never releases the control. Normally, an operator is supposed to have the responsibility to serve the customer, so he will not deliberately hold the control forever without giving a chance to the customer to gain control. At least, the customer can make such a request orally over the VoIP conversation. However, if the customer becomes the master, it is more possible that starvation occurs in case of some 'naughty' customers playing tricks on the operators. The following is a finite state machine showing the possible transition of control right between the customer and the operator.

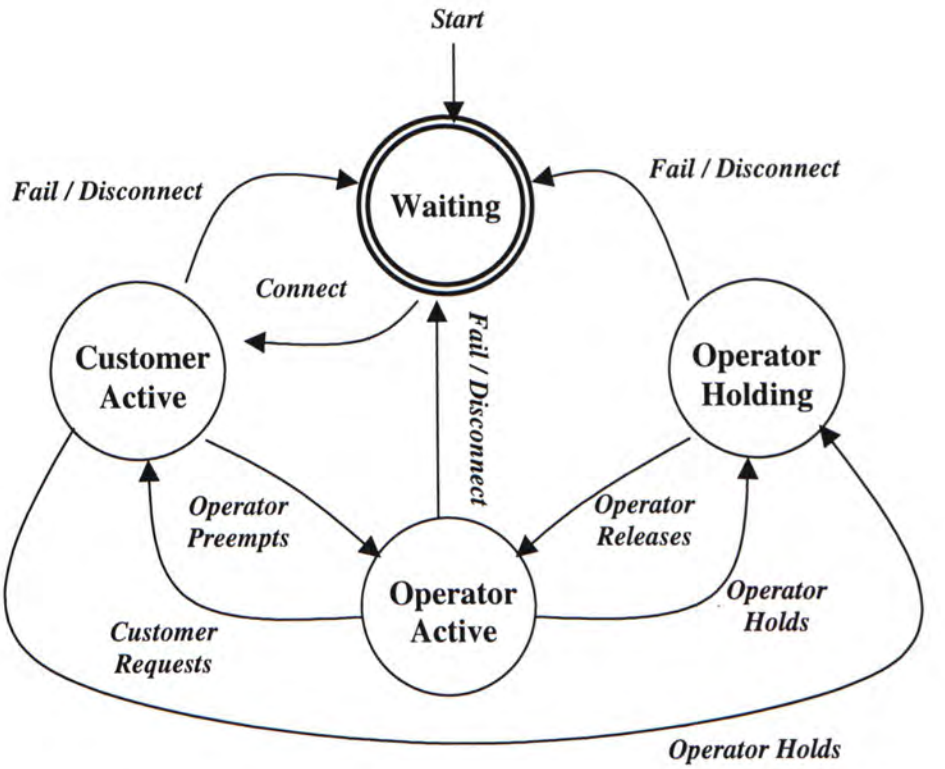


Figure 24 Transition of Control Right between a customer and an operator

Collaboration Facilities	Active (with privilege)	Passive (without privilege)
Tele-pointing	Enabled	Disabled
Whiteboard	Enabled	Disabled
URL Push	Enabled	Disabled
Form Elements	Any value change on the cell data will be pushed to the remote side	Data input locally will always be overwritten by remote push
Scrollbars	Enabled	Scrollbar positions will always be synchronized with the remote acitve side even though temporary scrolling to other position is allowed at the passive side

Table 1 Access Permission of the Collaboration Facilities in various Control States

The current collaborative browsing control protocol is designed to support a simple one-to-one communication (i.e. one operator to one customer) only. With such a small-scale networking environment, embedding a complicated control arbitration protocol into the collaboration mechanism only adds extra loading to the overall system but provides no obvious advantage. Thus, it is not considered.

3.2.4. The Hinting Mechanism for Collaborative Form Manipulation

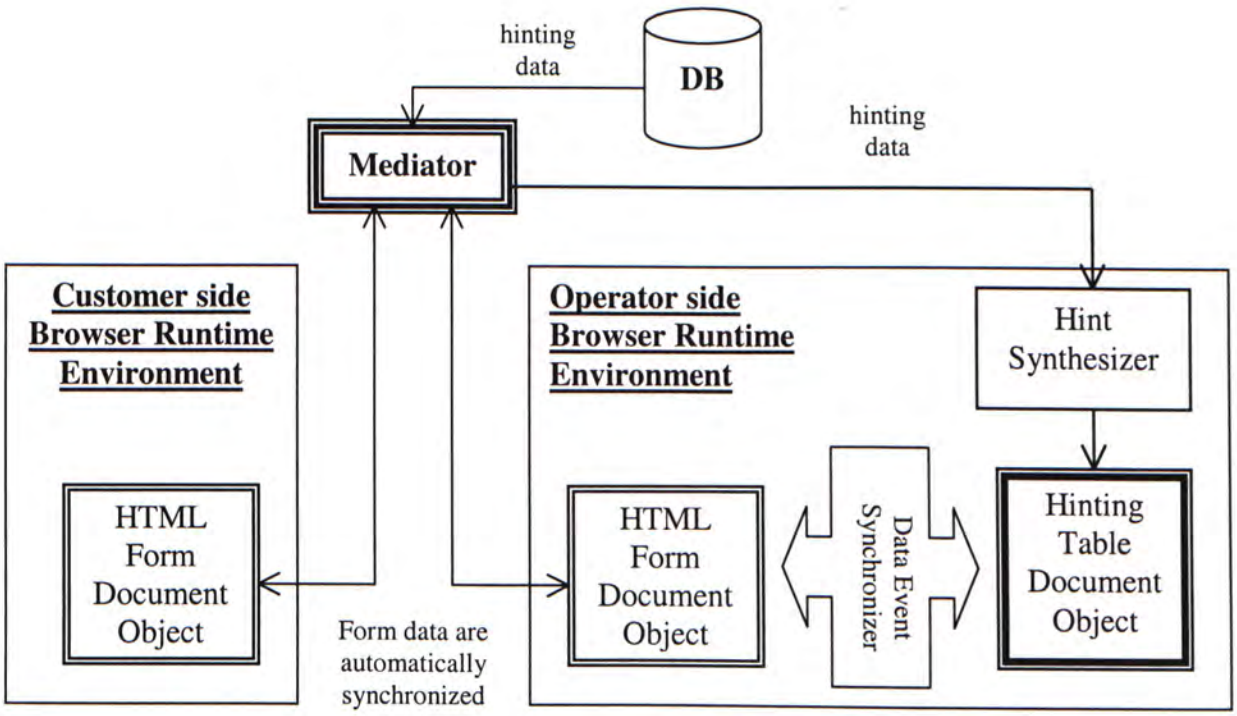


Figure 25 Software Architecture of Hinting

The working of the hinting mechanism is based on the software architecture shown in Figure 25. The mediator is responsible for retrieving hinting data from the backend database to the operator's browser as well as the synchronization of the form data between the browsers of the customer and the operator. Any user input

event or data modification in either the form fields of HTML form or the table fields of the associated Hinting Tables is synchronized on both sides by the Data Event Synchronizer. For example, when a particular table field is highlighted or focused, the HTML form page is automatically scrolled to show the corresponding form field on the screen for cross-referencing.

There are two HTML Form Document Objects, each models an HTML form in each of the two browsers. Any form data manipulation, such as automatic form data synchronization between the customer and the operator, acts on these two objects.

The role of the backend database is to provide meta-data, such as hinting data, to the mediator to synthesize the Hinting Tables for collaborative browsing.

During form collaboration, the operator can access the Hinting Tables by pressing a button on the control panel of his browser. Then, the hinting data corresponding to the form are retrieved from the backend database to the Hint Synthesizer by the mediator. The Hint Synthesizer then generates the Hinting Tables by instantiating and initializing the Hinting Table Document Object in the operator's browser. Now, the Hinting Table Document Object, representing the set of the Hinting Tables, traces the links and displays them. The Data Event Synchronizer is responsible for transforming the data in a form field to the data in the corresponding table fields in order to let the operator gain a better picture of the function(s) of the form data.

The implementation details, such as how hinting data are synthesized in the Hinting Tables, how data are synchronized between an HTML form and the Hinting Tables, will be discussed in Chapter 4.

3.3. *The Collaboration Mechanism for Collaborative IVR*

As described in the previous section, collaborative IVR supports the collaboration between the customer and the operator in a heterogeneous environment, in which the end terminals used by the customer and the operator are different. Since the customer uses a phone, which is of limited functionality when compared with a browser running on a PC, the collaboration methodology is slightly different from that of the original collaborative form manipulation.

Using the previous phone banking scenario as an example:

System : **Welcome to the Phone Banking IVR Service,**
Please press '1' for Account Manipulation, '2' for Financial Services.
Customer : **[Press 2]**
System : **Please press '1' for Balance Query, '2' for Fund Transfer, '3' for Deposit Renewal.**
Customer : **[Press 2]**
System : **Please enter the account number from which the fund is transferred.**
Customer : **[Press the hot key to contact an operator]**
<< Collaborative IVR Session Starts >>
Operator : Hi! You are in the Fund Transfer Service Node. Can I help you?
Customer : I'd like to make a query on the exchange rate of Hong Kong dollar to US dollar, but I've got lost.
Operator : I can get you to the corresponding service node. [Manually change the customer's current IVR traversal status to go to the service node for FOREX ratio query] Do you have any other problem?
Customer : No, thanks.
[Either the customer or the operator can end the session by pressing hot key (for customer) or button in the browser (for operator)]
<< Collaborative IVR Session Ends >>
System : Please enter the representation code of the foreign currency.
Customer : [Input the code through the phone keypad]
...

Figure 26 How Collaborative IVR works

During an IVR session, like phone banking, a customer may come across difficulties in addressing his desired IVR service, such as 'getting lost' in the IVR service tree, confusion of how to input the correct data through the phone keypad. To help, he can contact the call center operator by pressing a particular hot key during an IVR tree traversal. After that, a collaborative IVR session is established to contact the customer, working on a phone, and the operator, working on a browser.

On the customer side, the service is nearly identical to the traditional IVR service, in which the customer interacts with IVR system through the phone keypad. On the operator side, the operator can perceive the current IVR surfing status of the customer in form of an IVR tree displayed graphically as well as a set of HTML forms (Telephone Forms) representing the terminal service nodes of the IVR tree. The operator can help the customer by either directly traversing the IVR tree for the customer (i.e. set the customer to the desired function node arbitrarily) or tracing the IVR tree for supportive information (i.e. identify the key input sequence to access a particular service).

Similar to regular collaborative browsing, there exists a channel for voice communication **[11]** between the customer and the operator. However, there is no PC-to-PC communication. Therefore, the operator has to communicate with the customer and simultaneously manipulate the IVR operations for him through an HTML form interaction mechanism of the browser.

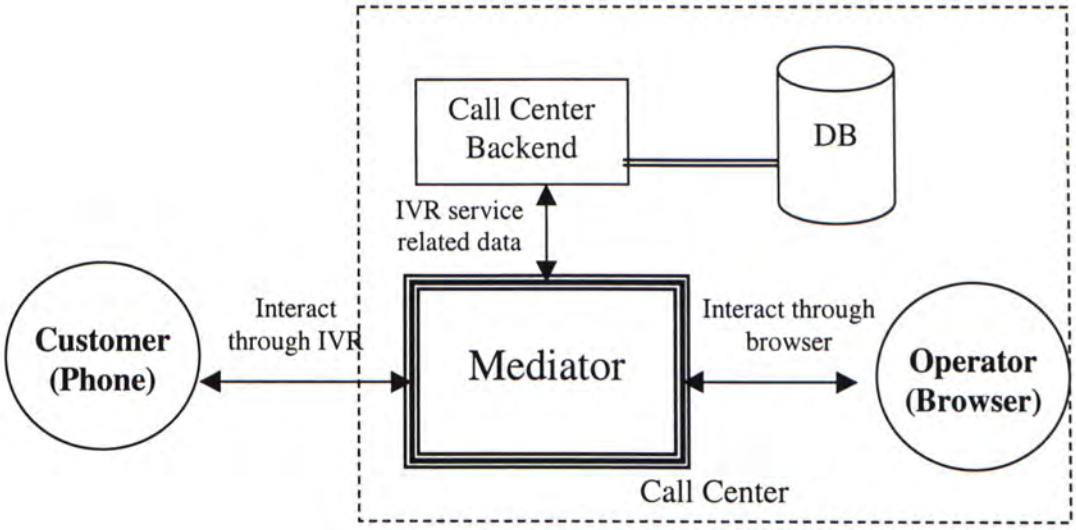


Figure 27 Software Architecture of Collaborative IVR

Figure 27 is the architectural design of Collaborative IVR system. The mediator plays an indispensible role for the entire Collaborative IVR process. It has three major tasks: conversion of IVR signals to the data conformable to the data structure known to the system; retrieval and output of appropriate Telephone Forms to an operator's browser corresponding to the customer's demand; and control of the IVR tree hinting mechanism.

The call center backend refers to the other call center modules, such as the Automatic Call Distributor, the IVR sub-system and some Voice-over-IP modules. It is necessary for the mediator to communicate with the call center backend when the mediator wants to change the current IVR tree traversal status upon an operator's request, like the arbitrary assignment of a service node to a particular IVR tree location, etc.

The database stores the data supplied by the customer during an IVR service. Besides, it stores the IVR tree information and the Telephone Forms.

Similar to a PC-to-PC collaborative browsing system, the Phone-to-PC collaborative IVR is also based on a three-tier architecture. Theoretically, the mediator based on a common backend architecture can be preserved and re-used for both prototypes. Also, the operator can keep using a unified browser interface to interact with customers, no matter whether he is manipulating HTML form data under collaborative browsing or IVR data under collaborative IVR.

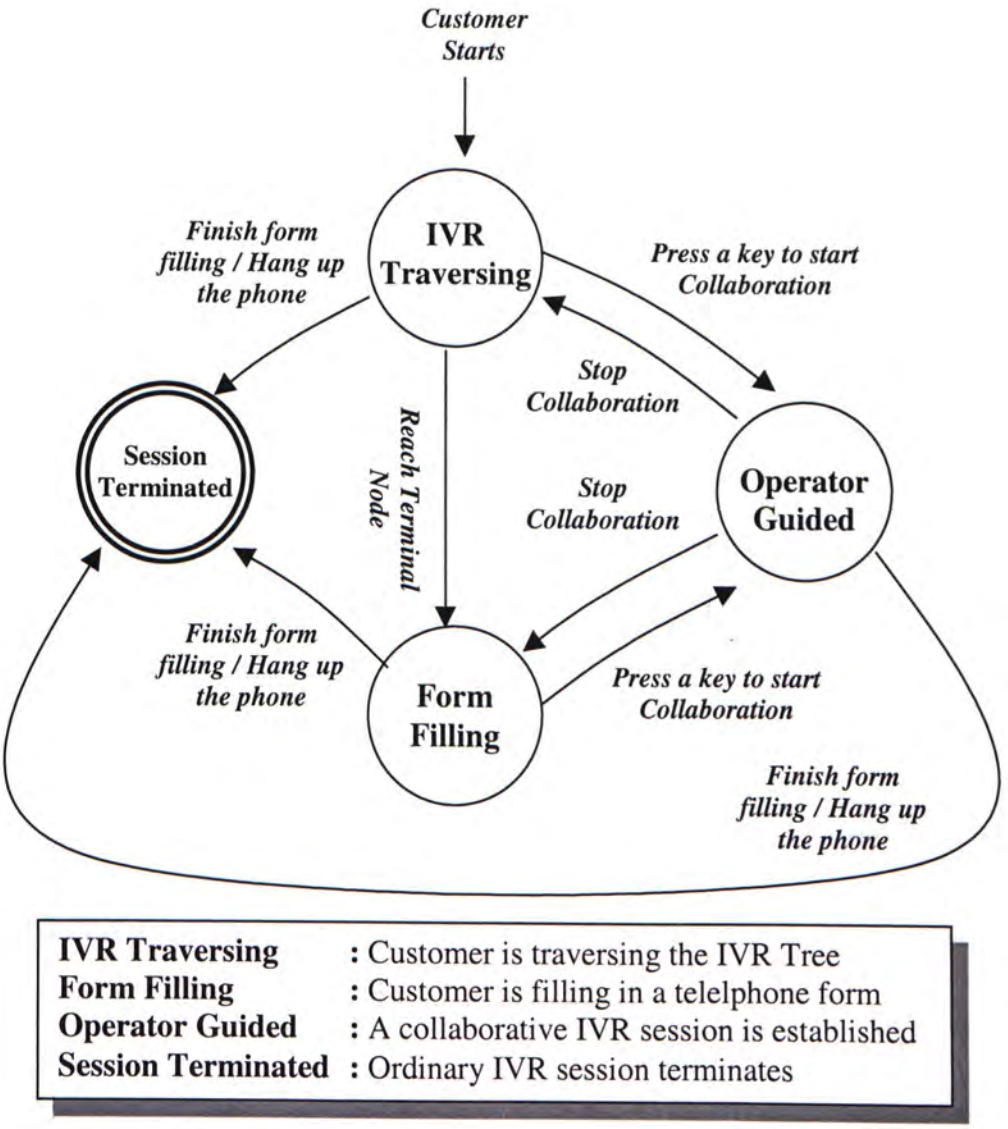


Figure 28 Transition of Operation States in Collaborative IVR

The above finite state machine depicts the collaboration flow between a customer and an operator. The operation procedure flow is shown in the following box.

1. A customer enters an ordinary IVR session once he has successfully made a dial-up connection to the IVR system of a call center.
2. The customer traverses the IVR tree to access a particular node for the desired IVR service.
3. During the IVR tree traversal, the customer can contact the remote operator at any time by pressing a hot key.
4. If so, the Operator Guided mode, in which the operator can assist the customer in accessing a particular IVR service, or inputting Telephone Form data, while the customer can describe the problem through the phone at the same time, is entered.
5. In the Operator Guided mode, any phone keypad input by the customer is neglected unless the operator enables the IVR system to receive the customer input. This arrangement prevents simultaneous entry of data when the operator is manipulating the IVR data for the customer while the customer is still inputting the IVR data for IVR tree traversal or filling in a Telephone Form.
6. The customer can leave the Operator-Guided mode by pressing another hot key at any time, say, the moment that the operator has finished the collaborative guidance.
7. After a customer has reached the target IVR terminal node, the customer is required to fill in a Telephone Form (Form Filling State).
8. He can, of course, contact the operator through a hot key while filling in the form. Alternatively, the operator can fill in the Telephone Form displayed in his browser for the customer.
9. Once the form filling process finishes, the final IVR service information, such as, some query results, will be output to the customer through the phone by text-to-speech synthesis or playing some pre-recorded audio files. After that, the entire IVR session terminates and the customer can hang up the phone.

Note: It is possible for a customer to hang up the phone to terminate a session at any time, no matter whether he is traversing the IVR tree or filling in a Telephone Form.

In collaborative form manipulation between a customer's PC and an operator's workstation, a set of Hinting Tables are provided to the operator for hinting the hidden data relationships of the form fields for better handling of the customer's problems during collaboration.

In collaborative IVR, a corresponding version of Hinting is necessary. The structural information of the IVR service tree is retrieved from the backend database and displayed at the operator's workstation. This tree information, as hints to the operator, enables the operator to perceive the currently IVR tree traversal status. Also, the operator can manually trace the IVR tree on his own to search the IVR customer's target service. The implementation details will be discussed in the next section.

4. Implementation

4.1. Shareable Document Object Architecture for Collaboration

4.1.1. Document Object Architecture

The goal of collaborative browsing is to let users collaboratively manipulate the web page contents in a distributed environment. Therefore, the document structure of the web page content plays a critical role in the implementation of the collaborative browsing. Conventionally, if we want to access the content of a particular text file, say, MS-DOS text file, the usual way is to retrieve the data one-by-one, line-by-line and page-by-page. Web page documents are different because they are well structured by a set of HTML tags. It turns out that we can have a higher level of document abstraction to favour the data access.

Inside a browser, all the web document data are organized under a Document Object Architecture (DOA) [41], which is object-based, slightly different from object-oriented. Each of the document data in an HTML page can be regarded as an object. Therefore, the content manipulation, such as layering HTML pages, getting the page URL for web page pushing, retrieving the FORM data for synchronization, inside each of the HTML page can be regarded as managing a set of JavaScript objects.

The object hierarchy starts from a top level HTML window object, which contains three mutually independent document objects (Tele-pointer Layer,

Whiteboard Layer and HTML Content Page) as its instance variables of the top level HTML window object. In particular, the HTML Content Page object contains a set of JavaScript objects representing the HTML document properties, like graphics, hyperlink, form data. Therefore, each of these document properties can be accessed through some instance variables, each of which, in turn, denotes a JavaScript Object related to the document properties, HTML Content Page object. As a whole, access to the document features, such as, form data, is based on object interaction under the DOA tree.

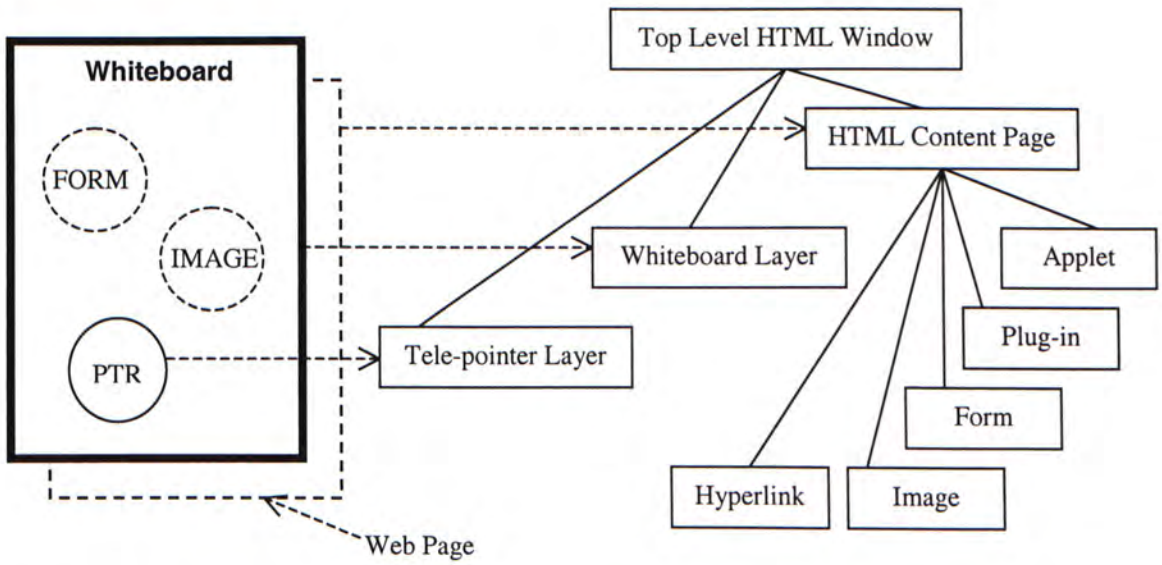


Figure 29 Tree Diagram representing the Object Hierarchy of various Web Content Data

In particular, the whiteboard is regarded as a layer object. However, different from other collaborative features, the internal implementation of the whiteboard layer object is based on a browser plug-in. It is written in native code and is, therefore, not a real JavaScript Object. Its internal work mechanism is explained in Section 4.2.

4.1.2. Generalizing to Shareable Document Object Architecture

As web contents are organized in the form of JavaScript Objects under DOA inside a browser, collaborative browsing can be achieved by sharing the object data between the customer side browser and the operator side browser. Therefore, from the view of the entire web document architecture, the original DOA must be shareable.

A Shareable Document Object Architecture (SDOA) is realized by supporting object data synchronization between the DOA of the customer side browser and the DOA of the operator side browser. This can be implemented by using a pair of Java Applets dedicated for data transmission. The two Java Applets appear as ordinary JavaScript Objects hosted in the customer side DOA and the operator side DOA respectively. During collaboration, when there is data modification in one of the JavaScript Objects, the DOA will automatically capture this event and signal the Java Applet to transmit the relevant feature data to the remote side DOA for feature synchronization. The Java Applet of the receiving side DOA then dispatches the received feature data to the target JavaScript Object, which, in turn, updates its object data.

For example, after a customer has input some data into a particular form field of a web document, the customer side DOA captures this data change event in the Form object and signals the Java Applet to transmit the newly entered form data (feature data) to the operator side DOA. After that, the Java Applet of the operator

side DOA dispatches the received form data to the corresponding Form object, which updates the corresponding form field with the received data.

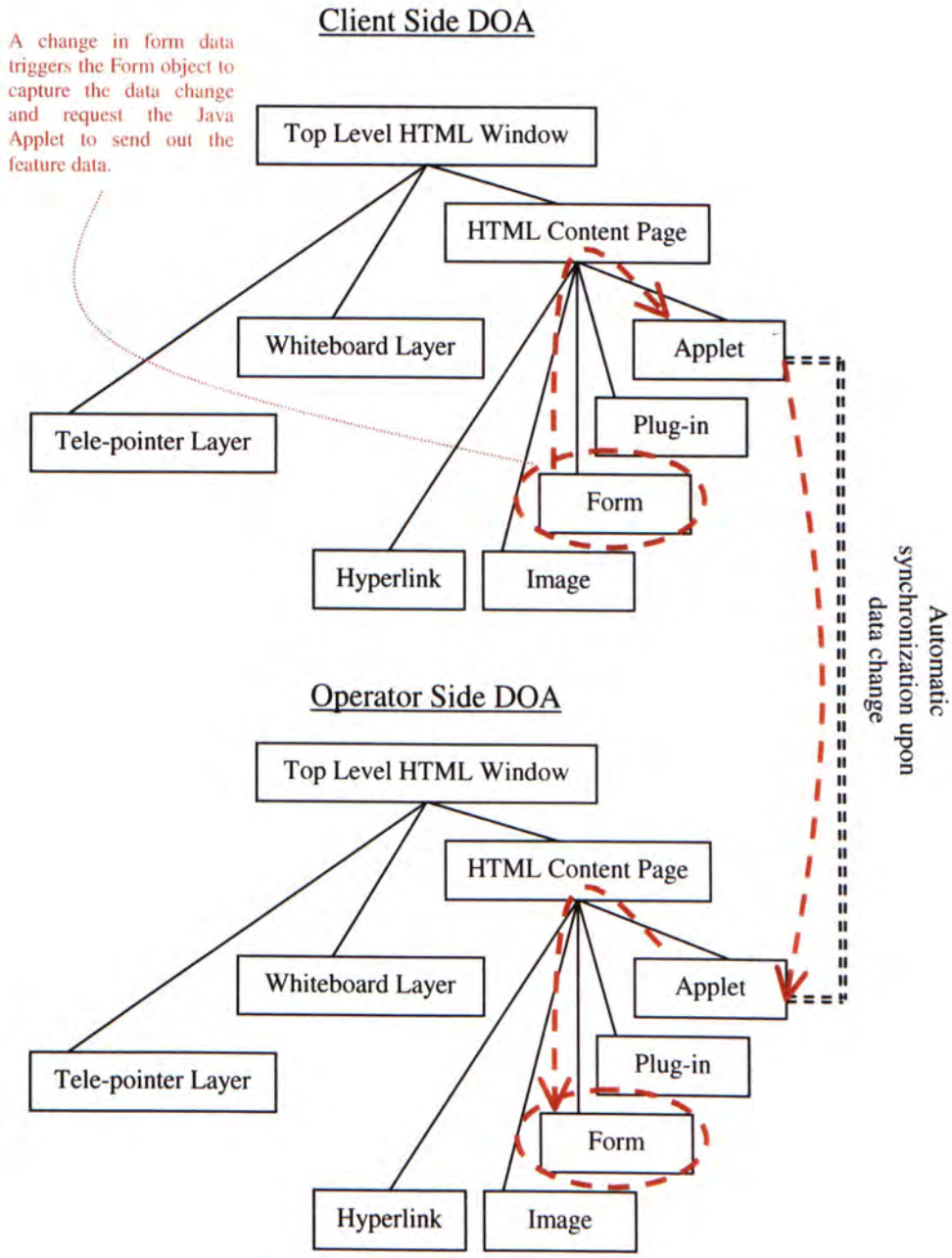


Figure 30 Shareable Document Object Architecture

4.2. Whiteboard Mechanism

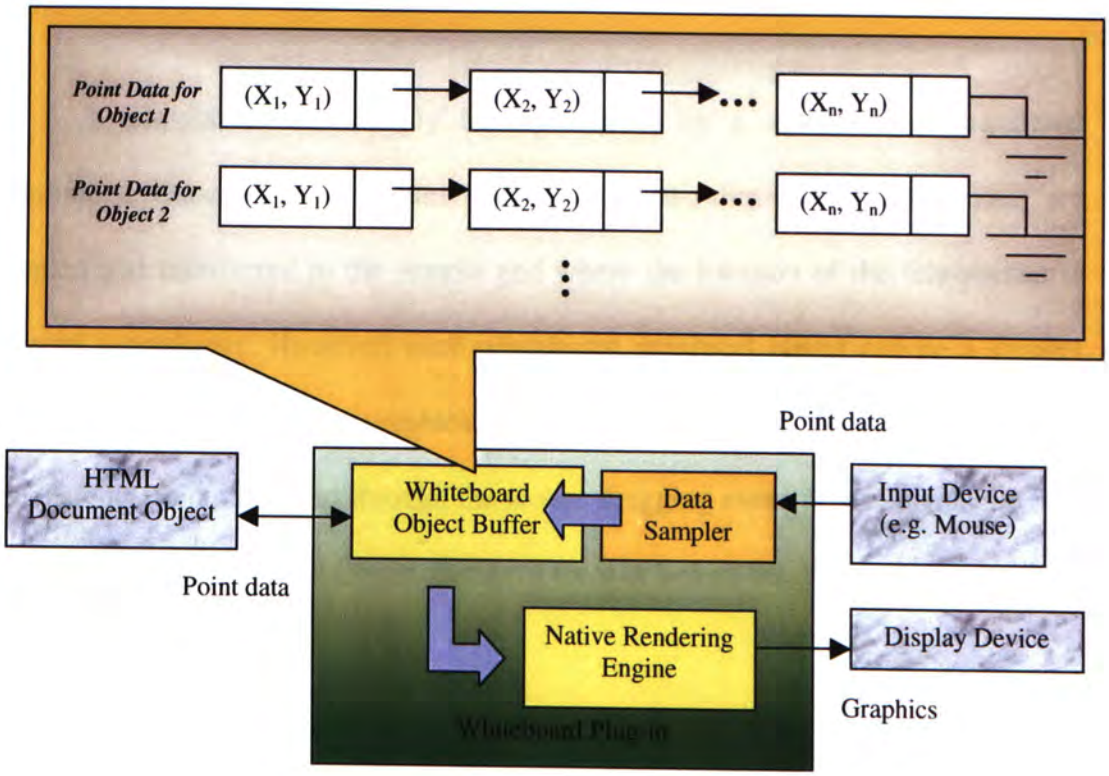


Figure 31 Whiteboard Architecture

The entire whiteboard plug-in can be divided into three main modules: Whiteboard Object Buffer, Data Sampler and Native Rendering Engine. The Whiteboard Object buffer stores the set of the whiteboard graphical objects drawn by the local user and the remote user. The Native Rendering Engine periodically retrieves the data inside the Whiteboard Object Buffer to synthesize the graphics onto the display. On the other hand, the Data Sampler continuously samples the mouse pointer position when the user is drawing on the whiteboard using a mouse. The sampled data are stored into the Whiteboard Object Buffer. At the same time, the sampled data captured from the local user who is now actively drawing on the whiteboard, are sent to the HTML Document Object, which, in turn, forwards the

data to the Java Applet for transmitting to the remote side browser. As a whole, the whiteboard plug-in is compiled in native code for faster execution.

A tele-pointer can simply be represented by a set of two-dimensional coordinates. Once a mouse pointer is moved, its two-dimensional coordinates are sampled and transferred to the remote end where the location of the tele-pointer is updated accordingly. However, each whiteboard graphical object can be a straight line, curve or polygon. A graphical object is represented by a series of two-dimensional points sampled through the mouse dragging event. These points are then stored into the Whiteboard Object Buffer in form of linked list data structure.

During the synchronization of the whiteboard objects in a collaborative browsing session, the point data of the whiteboard objects in the Whiteboard Object Buffer are transferred to the remote host in contrast to sending the complete bitmaps, saving much communication bandwidth.

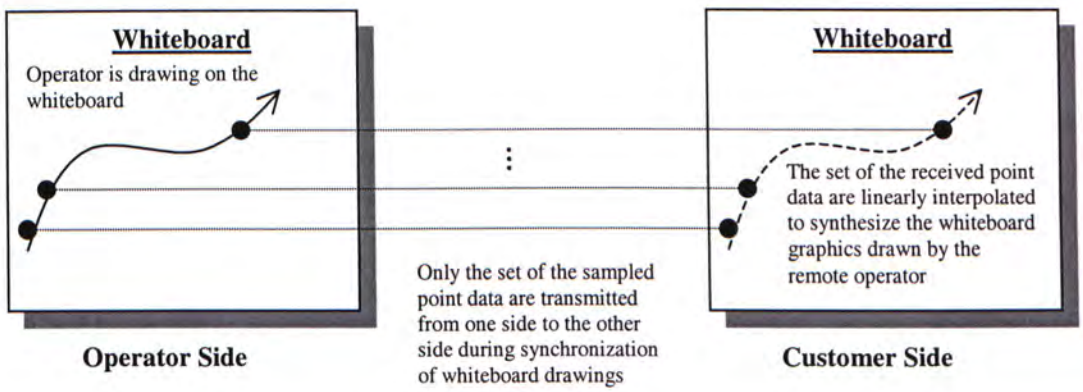


Figure 32 Point Data Synchronization for Shared Whiteboard

The data transmitted between the operator and the customer for whiteboard data synchronization are similar to those for tele-pointing as both processes only transmit a set of point data sampled locally to the other side. The difference is on the post-processing after the point data are received. For whiteboard service, the whiteboard plug-in will linearly interpolate the received whiteboard points and display the resulting graphical object. But, for tele-pointing, the received point data are used to update the current two-dimensional position of the tele-pointer, which emulates the mouse pointer on the remote side, one at a time.

4.3. *Packet Data Unit for Communication*

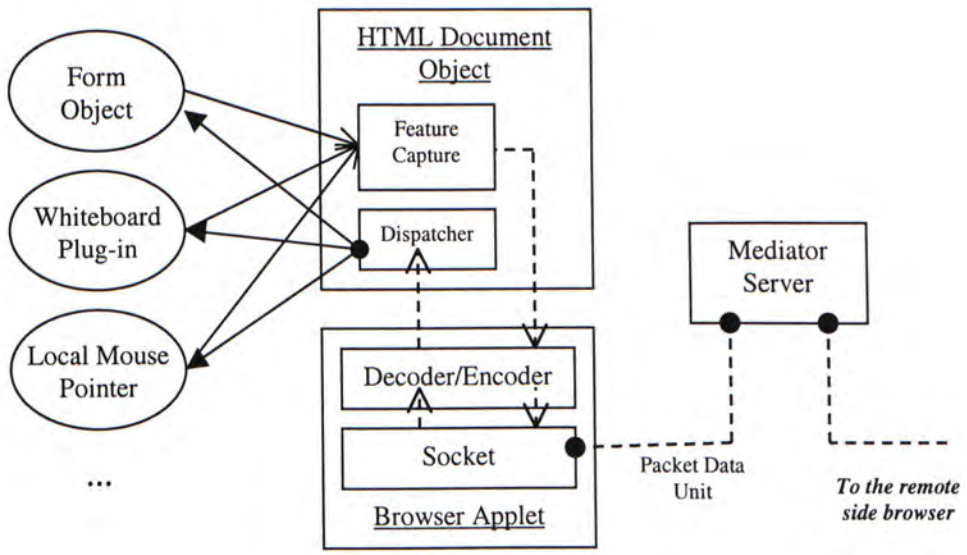


Figure 33 Packet Transmission for Feature Data Synchronization

In collaborative browsing, the web content features, such as the form data, are synchronized on both sides (operator and customer) for data sharing for collaboration. When the remote side browser commits a change in the shared

contents, a Packet Data Unit is generated and sent to the local side browser. Once the Packet Data Unit sent from remote side browser is received by socket module, the Decoder modules in the Java Applet breaks the data unit into three parts (CMD, P1 and P2) and forwards the decoded data to the dispatcher of the HTML Document Object. The dispatcher of the HTML Document Object, which is responsible for manipulating the web contents, then examines the Feature Type Identifier and routes the received feature data to the corresponding object for feature data update. For example, a packet contains feature update of a form field will be routed to the Form Object, which will, in turn, update the corresponding field by changing its instance variable.

Conversely, when there is a feature change, such as dragging a mouse pointer in the local web document, the Feature Capture module will record this change and forward the captured data to the Encoder of the Java Applet. After packing the Feature Type Identifier and the Data Parameters into a Packet Data Unit, the resulting Packet Data Unit will be sent out by the Socket module. A Packet Data Unit is marshaled into a character string by the Socket module before transmission.

The feature data are classified into several types and formatted into a series of packet data units before transmission by the Java Applet.

Packet Data Unit Format:

CMD	P1	P2

CMD : Feature Type Identifier
P1, P2 : Feature Data Parameters

Feature	CMD	Data Parameter 1 (P1)	Data Parameter 2 (P2)
Control Access	Access	access mode	-
Tele-pointer	Ptr	X	Y
Whiteboard	Wb	X	y
Clear Form	Clrfrm	-	-
Form Data	Form	form field index	form field value
Web Page Push	[URL]	X (Horizontal Scrollbar Position)	y (Vertical Scrollbar Position)

[URL]: The CMD field is particularly used to store the feature data (i.e. the URL of the web page) during web page push.

Table 2 Different Feature Types

Upon receiving the above packet data unit, the Java Applet examines the Feature Type Identifier to find out the type of the feature that should be updated. For example, if the operator drags the mouse pointer, a packet data unit with feature type 'ptr' and the corresponding geometric position (x, y) of the mouse pointer position is sent to the customer side browser. The Java Applet inside the customer side browser examines the Feature Type Identifier and then notifies the system for updating the position of the tele-pointer. Similarly, the other feature data are sent to the remote side according to the above format of the Packet Data Unit.

4.4. Bridging Different Software Components

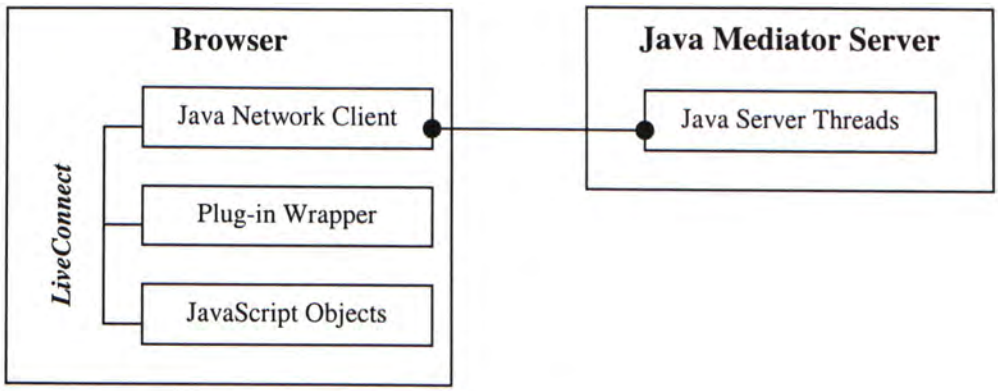


Figure 34 Modular Diagram illustrating the relationship between different software modules

The collaborative browser is built on an ordinary Netscape browser [42]. Therefore, users need not install a separate proprietary browser. The above diagram illustrates how different software components are linked together.

The Java Network Client class which is responsible for the data networking transfers the collaborative browsing data to or from the network. The other browser software modules, including the Plug-in Wrapper module, JavaScript Objects, have to communicate with this network module before they send or receive data.

The JavaScript Objects represent the HTML document data. They work under the DOA so that any access to the HTML document data has to be made through this module. In particular, we need to manipulate a JavaScript FORM object in order to access the HTML FORM data.

The Plug-in Wrapper class is created particularly for the whiteboard service. The data structures of the whiteboard graphical objects are stored and maintained by this module. Actually, it is a Netscape plug-in so that it can be regarded as written in native code. In this way, the browser can call platform dependent services provided by the operating system for faster execution speed, which is essential for real-time data capture and display of the whiteboard graphics.

The server side module refers to the mediator, which is made up of Java classes.

The communication between the above three modules inside the browser is based on an object wrapping [43] technique, called LiveConnect [44]. With LiveConnect, even though the plug-in is native code, it can be wrapped as if it was a Java class. This is called Java Wrapper. Correspondingly, LiveConnect further wraps each Java class as a JavaScript object, making all the modules inside the browser communicate through the ordinary JavaScript Object interaction.

To implement the Java Wrapper, the software routines between two different software environments, say, Java and native C++, should be firstly mapped accordingly so that any routine call from one environment, like Java, can invoke the corresponding routine implementation of another environment, like native C++.

Secondly, we need to transform the program data between the two environments for data consistency purpose. For example, Java boolean data type is supposed to have real truth value representation (i.e. true or false) while, in C++, on

which the plug-in coding is based, it is represented by an integer (i.e. 1 is true and 0 is false). Suppose we want to convert a Java boolean value into C++ boolean value. We have to convert the original value into a data type compatible to both software environments, say, a byte value, in Java environment. After that, we export the converted value to the target environment (C++), where we have to transform the imported byte value into the corresponding native integer type to represent the boolean value in C++. The conversion of other data types between Java and native C++ codes is similar. The following diagram is an example showing how the whiteboard plug-in module is wrapped as a Java class:

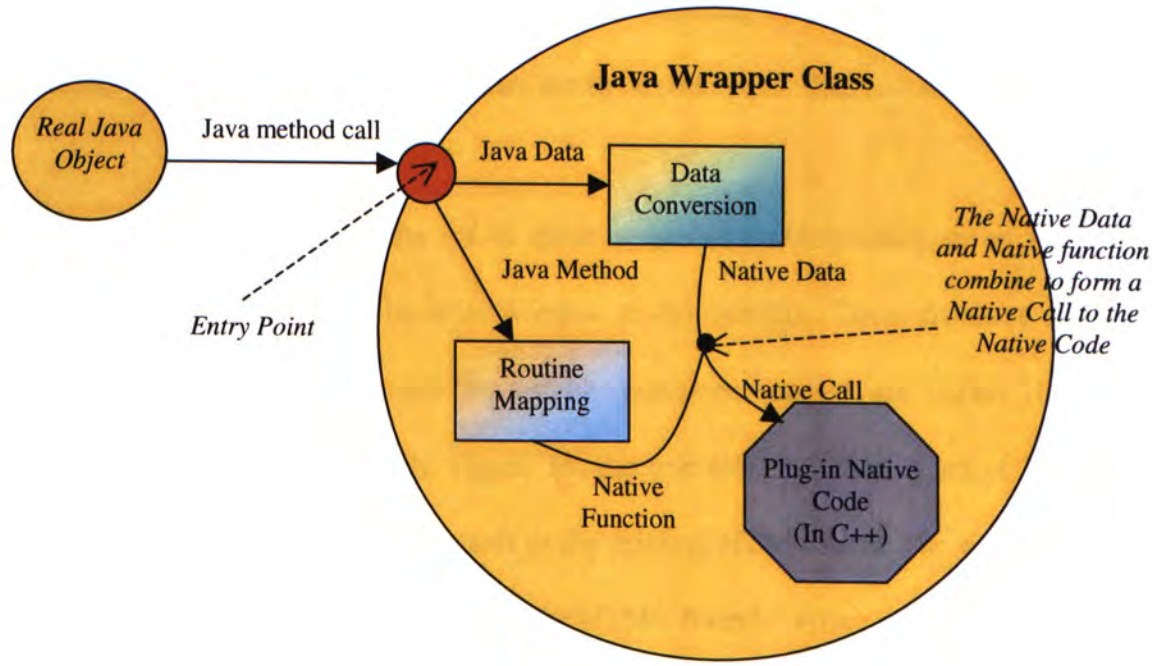


Figure 35 The Object Wrapping Technique used in LiveConnect

4.5. *Hinting Mechanism for Collaborative Form Manipulation*

4.5.1. Relating Form Fields to Table Fields

As described previously, an HTML form corresponds to a set of backend database tables because the form data entered by a customer are finally stored into the database tables. Since the data dependencies among the table fields should have been specified when creating the database, we can identify the data relationships among the form fields by linking the form fields to the table fields. After that, the operator can use the data relationships as shown in the table fields to help the customer fill in the form during a collaborative browsing session because data in the table fields reflect clearly how such data are to be processed. This is called Hinting.

To achieve Hinting, the set of database tables corresponding to an HTML form is visually shown in a browser window to the operator. These database tables which are used by the operator to provide customer support are called Hinting Tables. Eventually, as shown in Figure 36, the operator will have a set of Hinting Tables displayed on the screen, as well as the original HTML form. The operator can flexibly assist the customer in filling in the HTML form by either manipulating table fields inside the Hinting Table or entering data directly into the original HTML form. Through the Hinting Tables, the operator can see how the field values in the HTML form are mapped into those in the database tables and trace the data dependencies among the table fields in a step-by-step fashion.

In Figure 36, The Hinting Table relates a form and the corresponding database tables by mapping the data values in form fields into the table fields correspondingly.

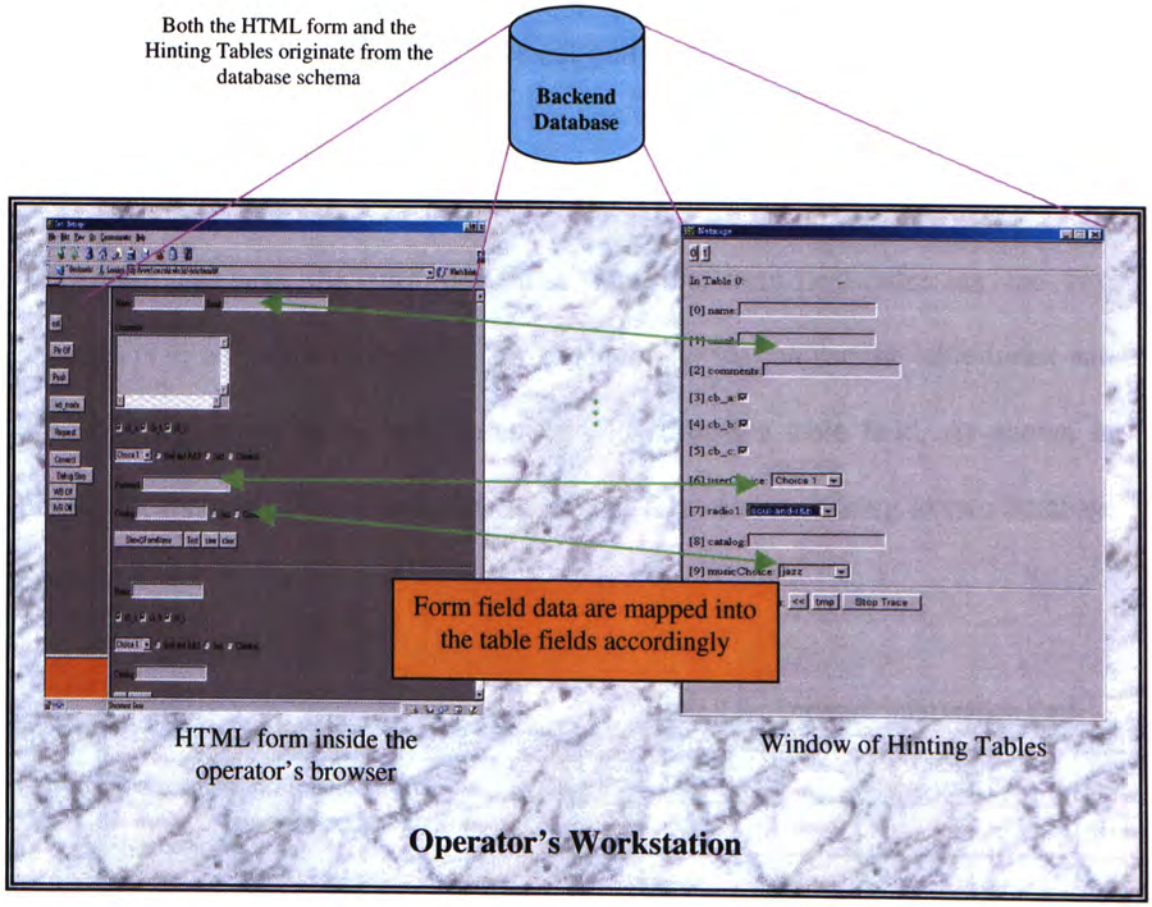


Figure 36 An HTML Form and the Hinting Tables in the operator's workstation

In general, it is possible to have different many-to-one and one-to-many mappings between the table fields and the form fields, as well as the simple one-to-one mapping. For example, one form field, say, a checkbox, may correspond to more than one table field in the database. To relate form fields to table fields, we use two address lookup tables. One table, **form2table**, is used to search the address(es) of the corresponding table field(s), given an address of a form field. The other table,

table2form, is used to search the address(es) of the corresponding form field(s), given an address of a table field. Since the relation between the form fields and the table fields, and the data dependencies among the table fields should have been specified according to the application context at the design time, the contents of the tables need not be separately synthesized. Here, we will discuss the methodology in using the table lookup for linking different parts of a form to the backend database tables.

We can use the form-part-index and the form-field-index as the two parameters to address a form field. Correspondingly, we can use the table-index and the table-field-index as the two parameters to address a table field. As shown in Figure 37, there are three form parts in the tax form corresponding to two database tables.

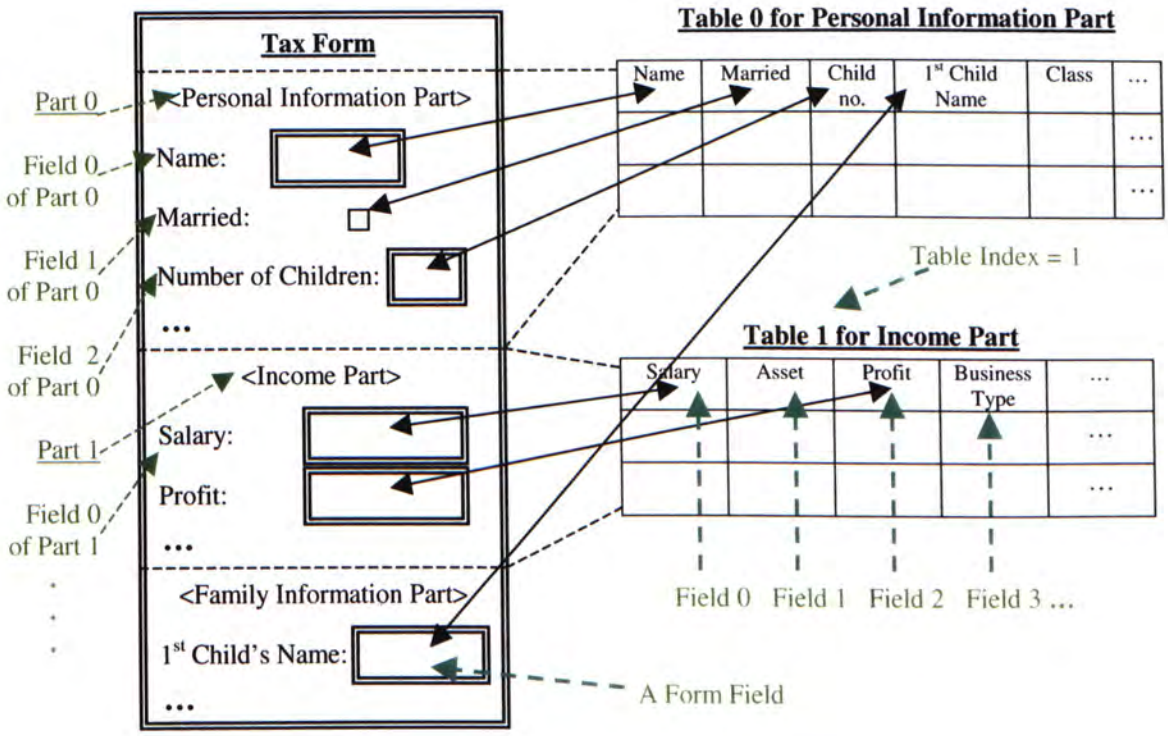


Figure 37 A tax form and the associated set of database tables

In the following explanation, we use (**a**, **b**) to represent an address of a table field/form field, where **a** denotes the table-index or the form-part-index and **b** denotes the table-field-index or the form-field-index.

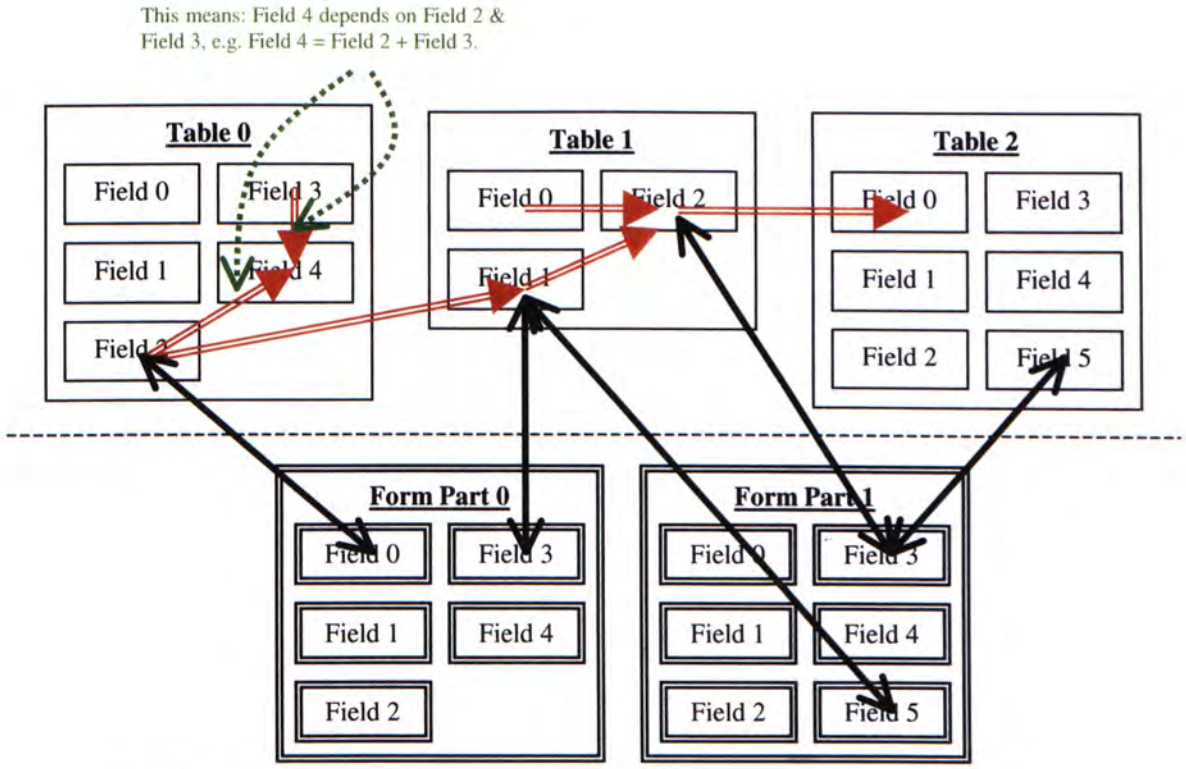
In Figure 38, the two form parts of an HTML form correspond to three backend database tables. Each of the black arrows relate a form field to a table field. In the three tables, the orange arrows represent the data dependencies inside the database. For example, suppose the data value stored in Field 4 of Table 0 is equal to the sum of the data values stored in Field 2 and Field 3 in the same table. This data dependency is graphically represented by drawing two orange arrows from Field 2 to Field 4 and from Field 3 to Field 4 in Table 0 respectively.

Two address lookup tables, **table2form** and **form2table**, are constructed to represent the graphical relationships depicted. They are implemented by a pair of two-dimensional arrays: **table2form** and **form2table**. **table2form[x][y]** refers to an array whose elements store the mapping information of the table field (x, y), where x is a table-index and y is a table-field-index. Similarly, **form2table[u][v]** refers to an array whose elements store the mapping information of the form field (u, v), where u is a form-part-index and v is a form-field-index. We will employ examples to illustrate the functions of these arrays. For example, a one-to-one relationship is shown by a black arrow, linking Field 2 of Table 0 and Field 0 of Form Part 0 in Figure 38. **table2form[0][2]** has been assigned a value (0, 0). **form2table[0][0]** has been assigned a value (0, 2). The other mappings between the form fields and the table fields are recorded in the same manner.

Not every mapping between form fields and table fields is one-to-one. Due to different processing requirements, it is not uncommon that a form field is matched to more than one table field value. For example, when the value of **form2table[1][3]** is retrieved, its content is two order pairs (1, 2) and (2, 5), which correspond to Field 2 of Table 1 and Field 5 of Table 2 respectively.

An element (x, y) in **table2form** may have values in columns T1 and T2. These values represent the dependency information among the table fields. T1 records all the outgoing dependency links from the table field (x, y) while T2 records all the incoming dependency links to the table field (x, y). To illustrate, let us look at element (0, 4) in **table2form**. This element has no value in T1 but (0, 2), (0, 3) in T2, meaning that its value depends on two table fields, namely (0, 2), (0, 3). Thus, tracing a dependency link from a particular table field (x, y) starts by traversing the outgoing links and incoming links of **table2form[x][y]**. The details of dependency tracing will be explained in the next section.

Graphical View



Array View

Table Index Table Field Index

Column T1 (Outgoing links) Column T2 (Incoming links)

tabel2form

…		
table2form[0][2] = (0, 0)	(0, 4) (1, 1)	…
table2form[0][3] = …	(0, 4)	…
table2form[0][4] = …	…	(0, 2) (0, 3)
table2form[1][0] = …	(1, 2)	…
table2form[1][1] = (0, 3) (1, 5)	(1, 2)	(0, 2)
table2form[1][2] = (1, 3)	(2, 0)	(1, 0) (1, 1)
table2form[2][0] = …	…	(1, 2)
…		
table2form[2][5] = (1, 3)	…	…

Form Part Index Form Field Index

form2table

…
form2table[0][0] = (0, 2)
…
form2table[0][3] = (1, 1)
…
form2table[1][3] = (1, 2) (2, 5)
…
form2table[1][5] = (1, 1)
…

Note: It is necessary to keep the dependency graph acyclic

Figure 38 How the two Lookup Tables correlate the Form Fields and the Table Fields

Since the two lookup tables are necessary for Hinting, the contents inside the tables, which are the mapping and dependency relationships, must be specified by the form designer when designing a form and the associated database tables.

As a whole, under the address table lookup mechanism, the form fields and the corresponding table fields can be associated and the related data dependency information can be obtained. The contents of the lookup tables will be transferred in form of the hinting data from the backend database to the operator side browser upon the startup of the hinting mechanism.

4.5.2. Hinting by the Hinting Tables

The Hinting Tables provide an operator with a simulated view of the backend database tables during collaboration. Consequently, an operator can perceive a set of form fields in the original HTML form and a set of table fields that represent the database schema associated with that HTML form.

During Hinting, the two lookup tables are retrieved from the backend database to the operator's browser. The Data Event Synchronizer, as shown in Figure 25, automatically retrieves the data values in the form fields of the HTML form and the table fields of the Hinting Tables by looking up the two tables. Thus, the operator can manipulate the form data represented in the table fields for customer support, instead of using the original HTML form. Data manipulation in the table fields is more convenient to an operator because data dependencies can help the operator to trace the relationships of the form data.

In a collaborative form filling session, when the operator presses the "Hint" button in the browser, a set of Hinting Tables are visually generated by the Hint Synthesizer, which is shown in Figure 25, according to the database schema information in the hinting data. The user interface of the Hinting Tables is shown in Figure 39.

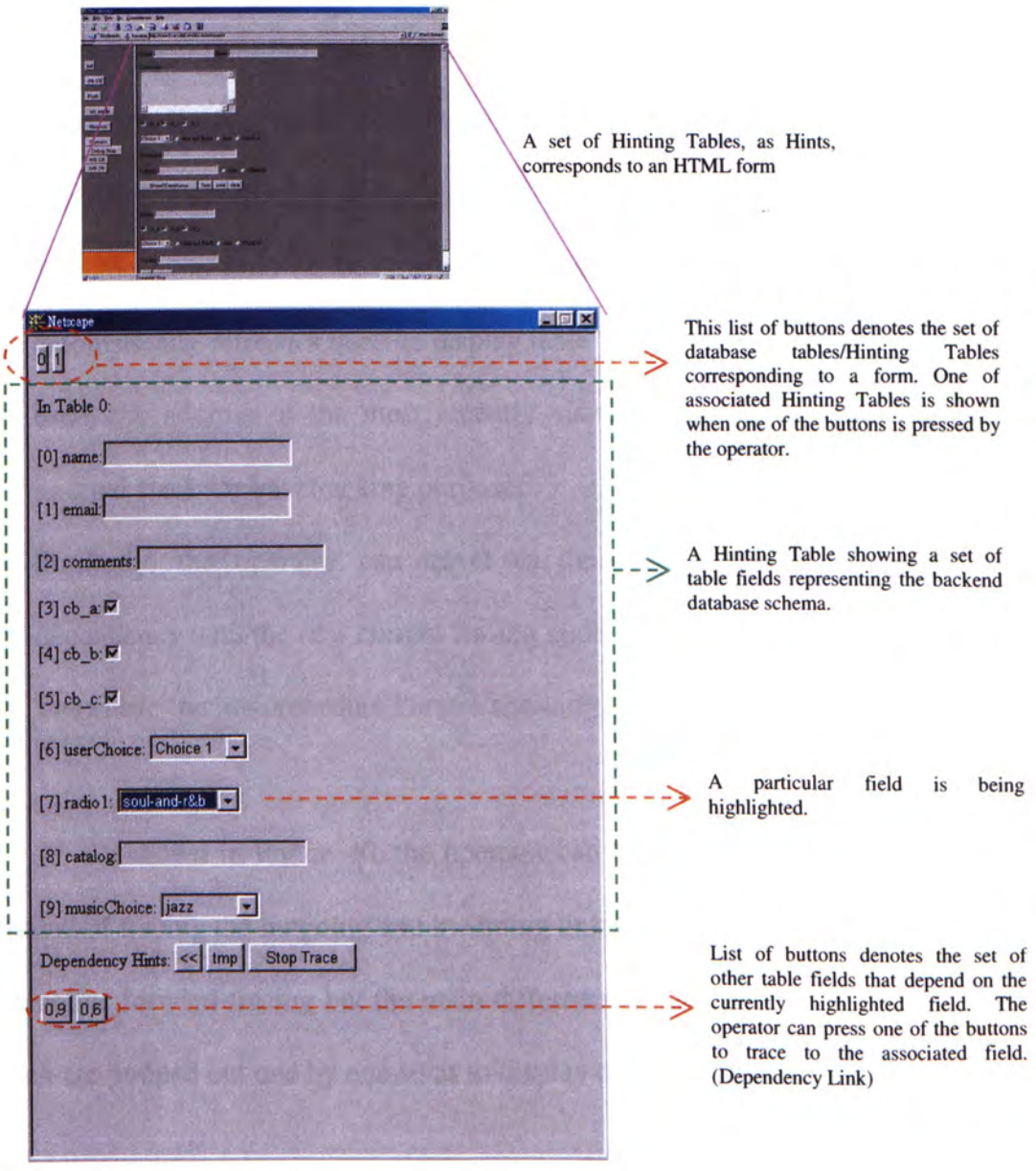


Figure 39 The Window of Hinting Tables

As form fields and table fields are related by the two lookup tables, the data relationships among the form fields of an HTML form can be identified indirectly by tracing the corresponding table fields in the Hinting Tables. For example, the steps of tracing the dependency link from a particular table field in Figure 38 are:

1) Suppose that an operator has highlighted field 2 of table 1 (i.e. (1, 2)) in a Hinting Table, this implies the tracing starts from field 2 of table 1. From there, we locate three other table fields (i.e. (1, 0), (1, 1) and (2, 0)), which have direct dependency relationship with the table field (1, 2).
2) Say, the table field (2, 0) is selected by pressing a button. The Hinting Table automatically refreshes itself to display table 2 and highlight field 0. At the same time, the address of the most recently visited entry (1, 2) is pushed into an internal stack for backtracking purpose.
3) Similarly, the operator can travel to the other table fields having direct dependency with the new current tracing node by clicking one of the buttons.
4) The entire tracing procedure iterates similarly until the operator stops tracing.

As shown in Figure 40, the operator can backtrack along his tracing action, instead of tracing the outgoing and incoming links. This procedural flow is similar to the above forward tracing but the main difference is that the contents in the internal stack are popped out one by one so as to display the previously visited table fields.

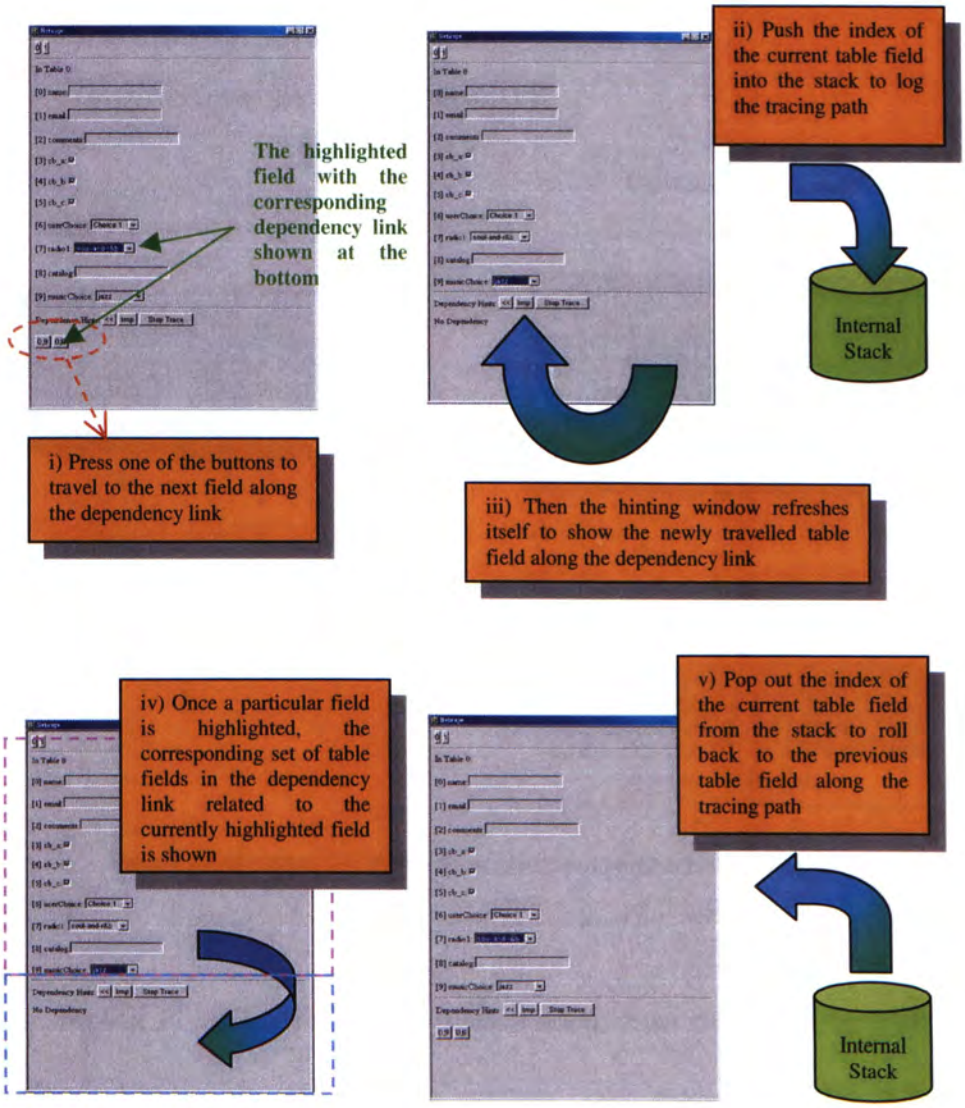


Figure 40 Tracing the data dependency links

4.6. Collaborative IVR

4.6.1. Using Mediator for Collaborative IVR

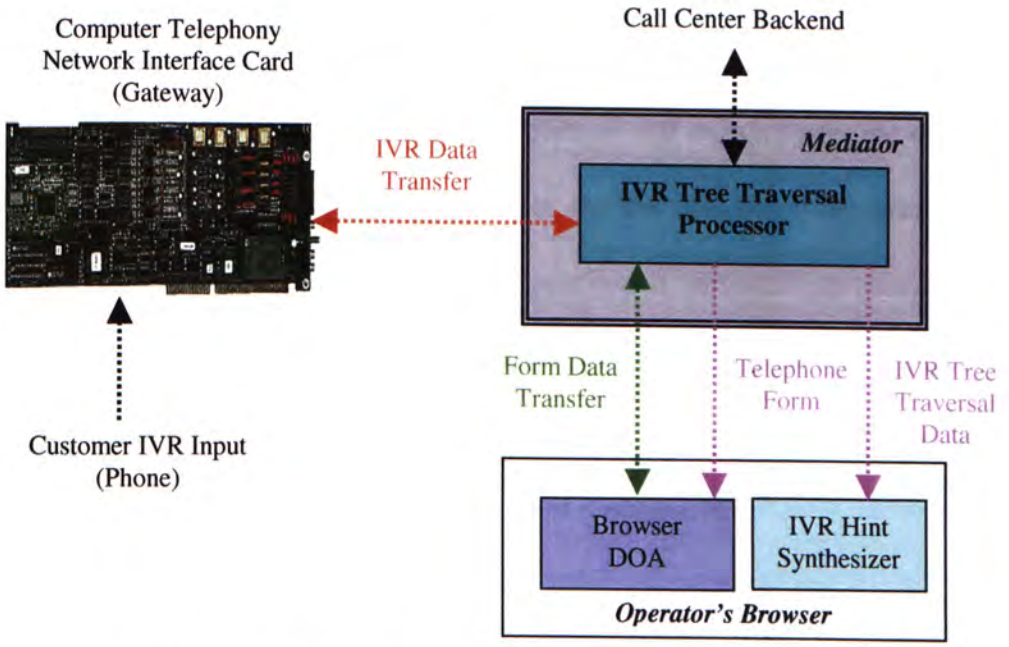


Figure 41 Modular Diagram inside the Mediator for Collaborative IVR

Figure 41 shows how the mediator transforms the customer IVR data into Telephone Form data so that an operator can manipulate the IVR data for collaboration.

In an IVR session, all IVR data input by a customer via a phone are captured and converted into the digital data by the Computer Telephony Network Interface Card [45], which is installed in the call center gateway. The converted IVR data are transferred to the mediator, which runs in the call center gateway computer also. The IVR Tree Traversal Processor (called the Traversal Processor in the subsequent discussion) possesses all the IVR service tree information retrieved from the call center backend and controls the IVR service traversal flow of a customer in the entire

IVR session. For example, any IVR request from the customer is processed and responded.

When a customer enters a collaborative IVR session, the Traversal Processor transforms the IVR data into form data for the operator's browser. If the customer is still traversing the IVR tree, the Traversal Processor will only transfer the customer's traversal history to the IVR Hint Synthesizer for the operator's browser. When a customer has selected a service, i.e. has reached a terminal service node, the Traversal Processor will transfer the corresponding Telephone Form to the Document Object Architecture (DOA) of the operator's browser, as well as transferring the traversal history to the IVR Hint Synthesizer.

As a whole, the Traversal Processor is responsible for transforming the IVR request-response interaction mechanism into the Form based request-response data exchange mechanism in a collaborative IVR session.

4.6.2. Concept of Telephone Form

In contrast to the traditional phone-request-system-response call center system, a Telephone Form is constructed for a terminal service node to handle the user-input interaction [46], [47] in collaborative IVR.

A Telephone Form can be regarded as an ordinary HTML form, which contains a set of widgets, such as text fields, checkboxes, selection list or radio

buttons, for inputting data. However, Telephone Form is used by an operator to manipulate the IVR service data for a customer.

For example, suppose that a banking service — travel insurance application is available as an IVR phone service, the corresponding IVR service flow can be regarded as sequentially asking a customer to input the required data one-by-one through a phone keypad. However, it turns out that we can use a Telephone Form, to represent all the data fields that are input by customer. This allows an operator to visually view the sequential IVR data entry process in a flat manner.

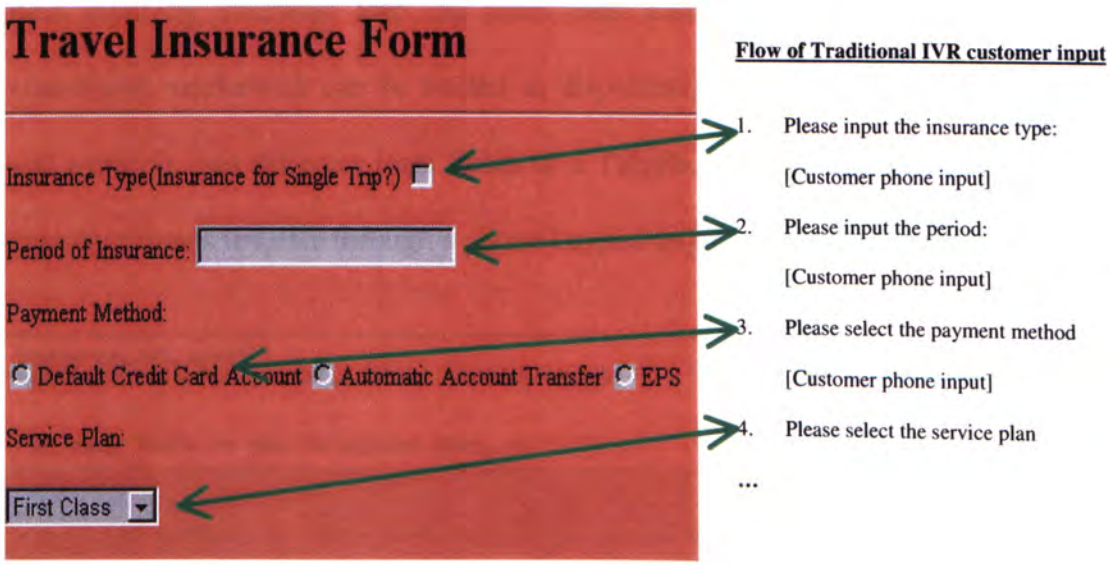


Figure 42 A Telephone Form and the corresponding IVR Input

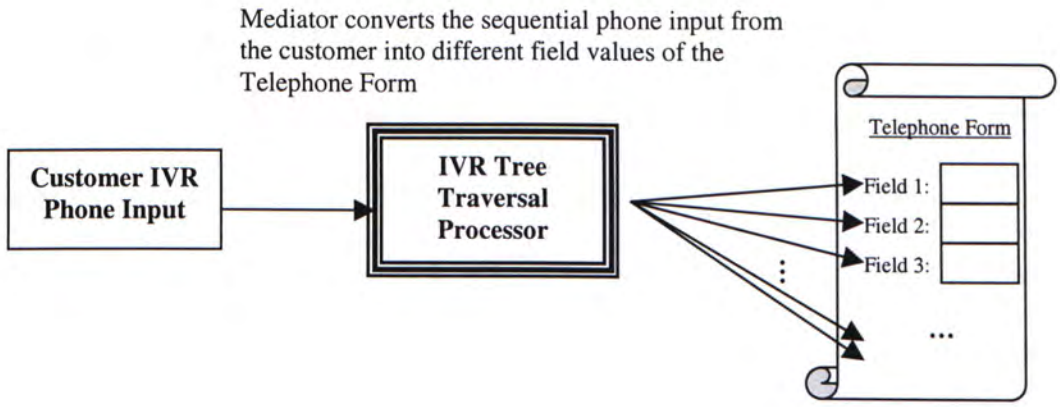


Figure 43 The role of the Mediator in supporting Telephone Form

The Traversal Processor inside the mediator is also responsible to convert the data input by customer into the form field data accordingly. The entire data conversion mechanism can be treated as dispatching a sequence of numeric values and symbols into different form fields of a Telephone Form one-by-one because a customer inputs the data through a phone keypad sequentially.

```
Traverse_IVR_tree();
Retrieve_telephone_form(terminal service node);

For each field in the Telephone Form, do {

Prompt the customer to enter data by playing an audio message file;        To customer

Switch (form field type) {
    Case "Textfield":
        Get & Enter customer input into the field until '#';
    Case "Selection List (Single)" or "Checkbox":
        Get customer input;
        Select_item(customer input);
    Case "Selection List (Single)" or "Checkbox":
        Get number of selected items;
        Get customer inputs;
        Select_items(customer inputs);
    Case "Radio Button":
        Get customer input;
        Toggle_radio_boolean(customer input)
  }                                                                        From Customer

}
```

Figure 44 The data conversion process

To dispatch the customer's data to the form fields correctly, the Traversal Processor needs to parse the data according to the type of a corresponding form field. There are four major types of form fields: text field, selection list, radio button and checkbox. For a text field, the data input from a customer are automatically entered into the field until the symbol '#' is keyed. For a selection list or radio button, a customer is given a list of items for selection, the desired item can be selected upon keying the corresponding item number in the phone keypad. In case of a multiple selection in a selection list, a customer has to enter the number of selections before pressing the phone keys to select the items. For a checkbox, a customer only needs to press a key once to input the Boolean value. Figure 44 shows the skeleton of the data conversion process inside the Traversal Processor.

With the Telephone Form, the three-tier call center architecture can be re-used for collaborative IVR, as well as the collaborative browsing. The operator can re-use the browser to manipulate the IVR data represented by a Telephone Form while the mediator can be re-used to transform the IVR data into the Telephone Form data. Also, the low level primitive IVR data, consisting of numerical values sequentially input by the phone keypad, can be transformed into high level complex form field data, which are more meaningful for the operator to understand the IVR data relationships. Lastly, with the use of some phone-based data input methods [48], [49], more complicated data types, such as alphanumeric strings, can be input from a phone keypad into a Telephone Form directly.

4.6.3. Hinting for Collaborative IVR

There is Hinting for collaborative form manipulation in the PC-to-PC collaborative browsing environment. Since the Telephone Form is a kind of HTML form, it is possible to derive a similar hinting mechanism specially to further enhance IVR collaboration, too.

Hinting in Collaborative IVR can be realized by transferring the IVR tree information stored at the backend database to an operator's browser. The complete IVR tree is graphically displayed in a hinting window to an operator as IVR hinting. In general, this IVR tree, or called IVR hint, provides the operator with a global view of the entire IVR service access methodology for accessing various IVR service nodes more easily. Based on the IVR hint, the operator can perceive and alter the IVR tree traversal status of a customer when necessary.

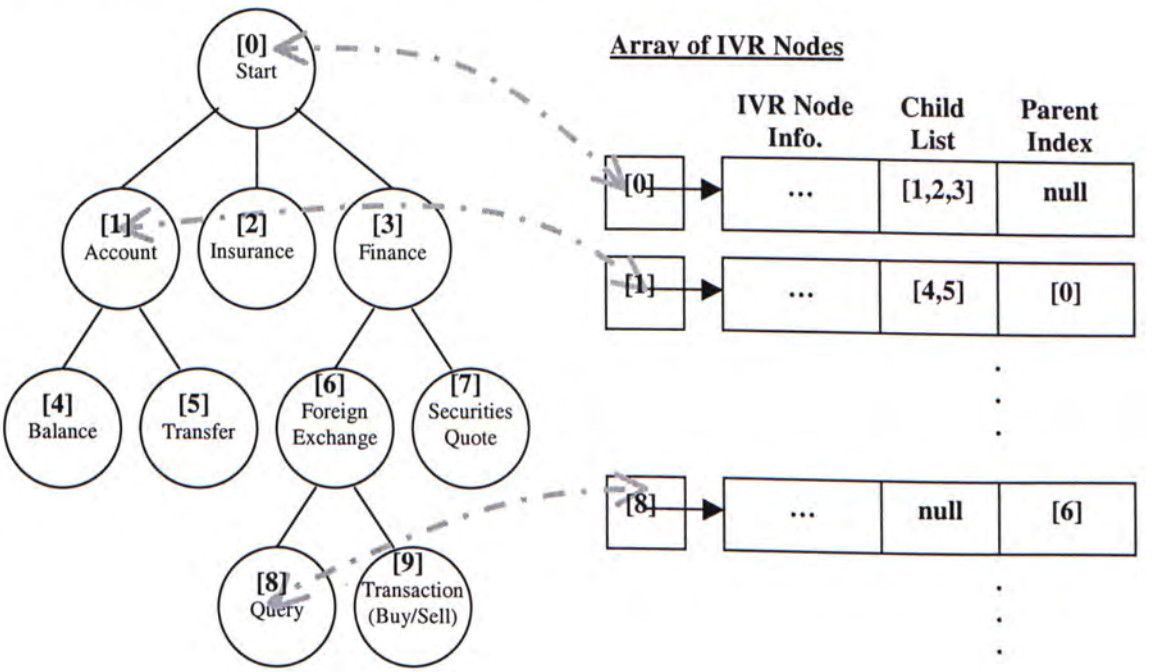


Figure 45 Array Representation of the IVR service tree for phone banking service

In Figure 45, the data structure shown on the right hand side is the array representation of the IVR tree on the left hand side. Each array element corresponds to a tree node. There are three segments in each array element. The first segment stores the IVR node information, such as the class of the IVR services the node belongs to. The second segment stores the array indices of the child nodes attached to the current node. The third segment stores the array index of the parent node.

Once a collaborative IVR session starts, the array of the IVR tree nodes, as the IVR hint, will be automatically transferred to the operator's browser from the backend database for the display of the IVR tree in the hinting window. Then a graphical IVR tree diagram is generated in the hinting window so that the operator can arbitrarily traverse to a particular tree node for the customer by a simple mouse click. Alternatively, the operator can trace the path from the root node to a particular terminal service node using the hinting window.

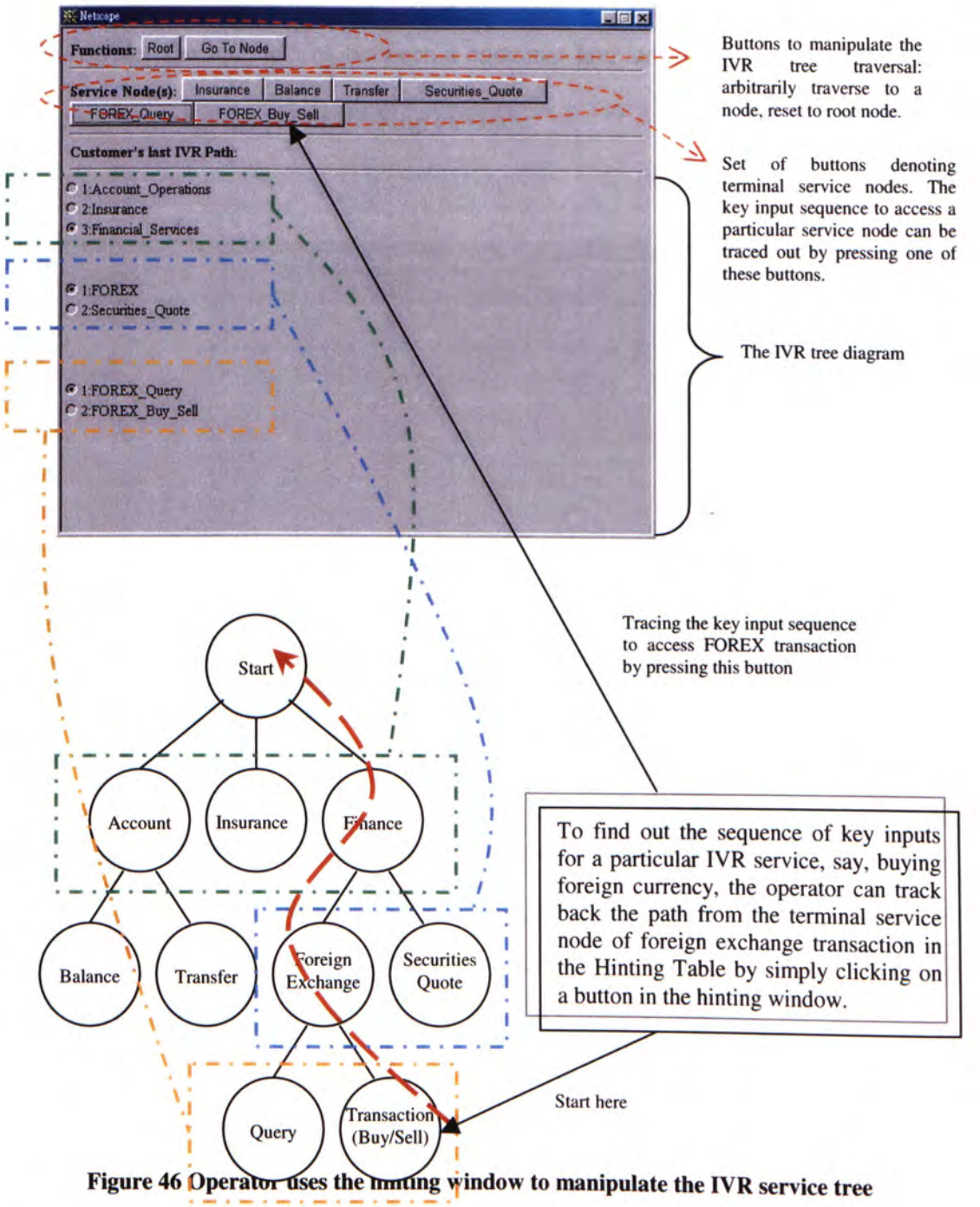


Figure 46 Operator uses the hinting window to manipulate the IVR service tree

When an operator arbitrarily traverses to a terminal service node for a customer in a collaborative IVR session, the corresponding Telephone Form will be displayed in the browser window of the operator's workstation so that the operator can assist the customer in filling in the Telephone Form at the customer's request. As a result, instead of keying the data by the customer, the operator can fill in the

different fields of the Telephone Form for the customer. The following dialogue demonstrates the interactions between a customer and an operator in a collaborative IVR session, using the phone banking example:

System	**: Welcome to the Phone Banking IVR Service, Please press '1' for Account Manipulation, '2' for Financial Services.**
...	
System	**: Please input the representation code of the foreign currency for query.**
Customer	**: [Press the hot key to contact an operator]**
<< Collaborative IVR Session starts >>	
Operator	: Hi! How can I help you? You are trying to query the exchange ratio of a foreign currency?
Customer	: I'd like to make a query on the exchange rate of Hong Kong dollar to US dollar, but I've forgotten the corresponding representation code.
Operator	: Ok. The code is 12345. I can input it for you also. [Input '12345' into the corresponding field of the Telephone Form & close this session]
Customer	: Thanks.
<< Collaborative IVR Session Ends >>	
System	: The exchange ratio of Hong Kong Dollar to US dollar is…
...	

Figure 47 Using Telephone Form in a Collaborative IVR session

4.7. System Integration

There are three parts for the entire digital call center system: Automatic Call Distributor (ACD), Collaborative Browsing and Voice-over-IP. This thesis focuses on the collaborative browsing system, which has to be integrated with the other two parts for a complete digital call center solution.

The ACD is responsible for delivering the address information of the operator's workstation and the customer's PC into the matching table of the mediator

for the initialization of a collaborative browsing session, as well as routing a suitable operator to the customer during call setup. As shown in Figure 48, when an operator and a customer would like to start a collaborative browsing session, their browsers will submit their requests to the mediator respectively. The mediator will enable the collaborative browsing session after successful lookup of their address information in the matching table. Otherwise, the connection request will be denied by the mediator.

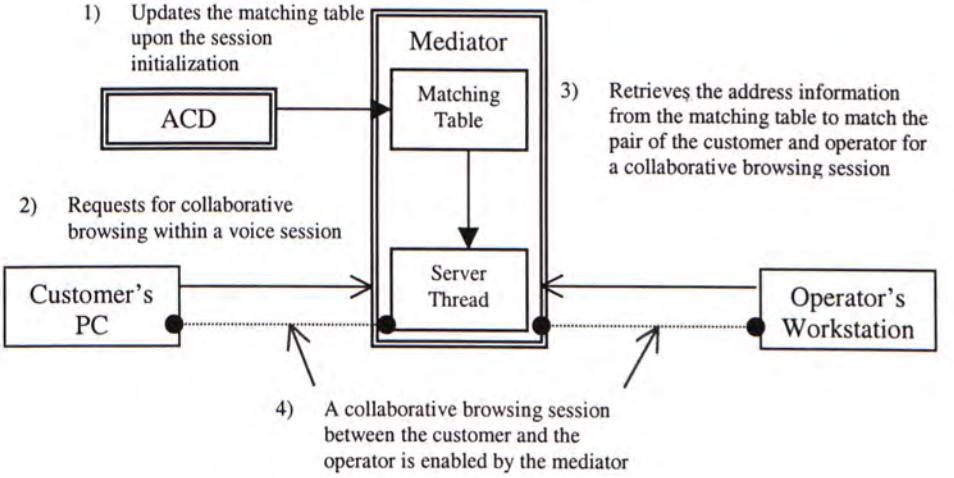


Figure 48 The role of ACD in the startup of a collaborative browsing session

The role of the ACD system on the Phone-to-PC collaborative IVR system is slightly different. In an IVR tree, the tree nodes of similar service types are grouped together to form a branch, representing a single service group. Call center operators are assigned into different operator groups, each of which handles the customer support related to one service group only. In other words, each operator group is responsible to handle customer support originated from a particular branch. For example, in Figure 49, an operator group dedicated for account related services in phone banking has to handle any customer request originated from the whole branch 'Account' in the IVR tree.

Once a customer presses a phone button to request for contacting a call center operator, the ACD can route the customer request to a particular operator of the suitable operator group according to which IVR node the customer is currently in based on the traversal history.

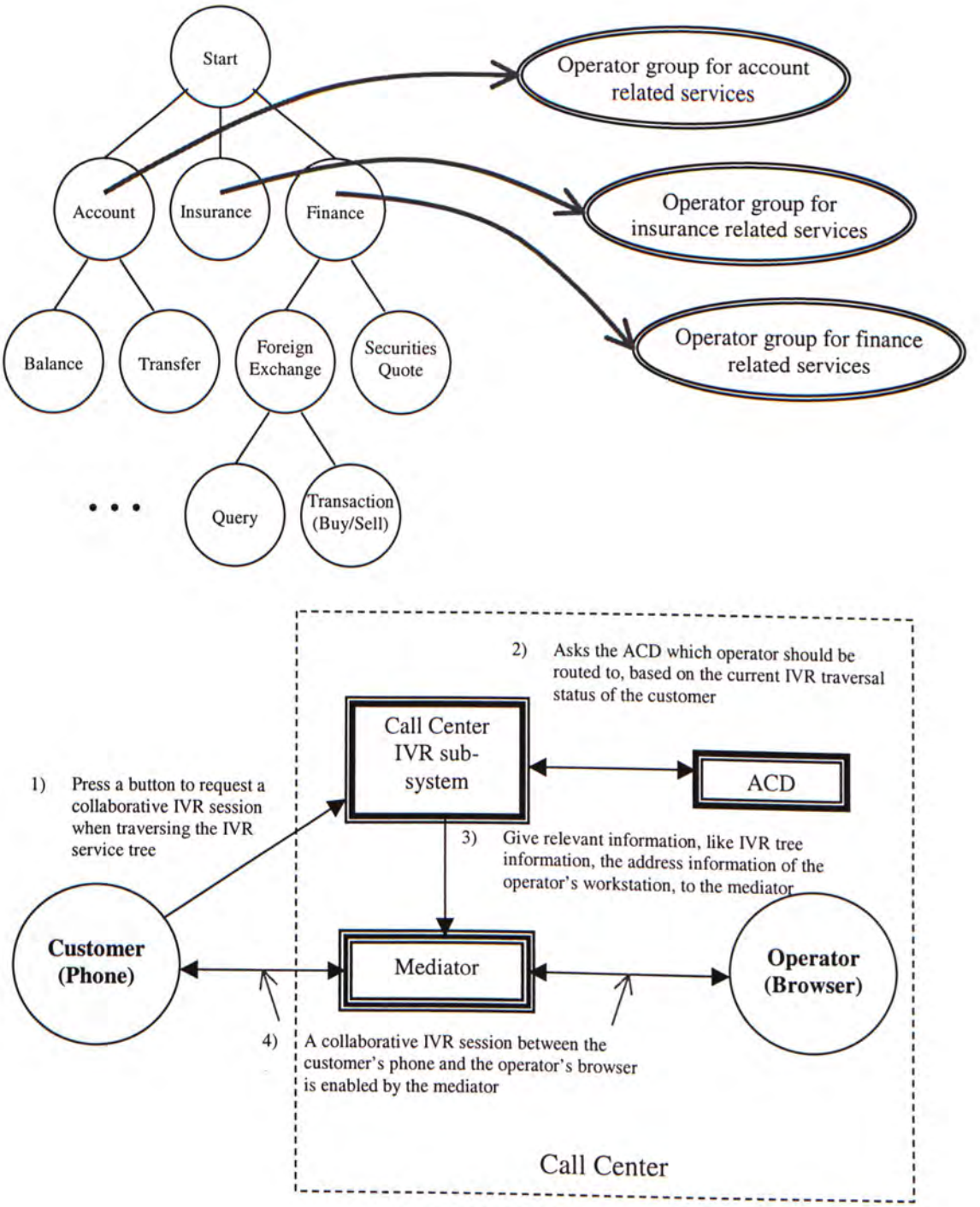


Figure 49 The role of ACD in a collaborative IVR session

5. Performance Evaluation and Experiment Results

5.1. Optimizing the Transmission Methodology

The VoIP and other data communication channels are based on TCP. It was found that the way we pack and deliver the data packets affects the communication delay. The original way of the TCP packet transmission is tailored for bandwidth efficiency. When there are data stored in the output buffer in the sending side, only the first byte of the buffer data is sent to the receiver side. The output buffer continues to accumulate the incoming data. Until the first byte sent out is acknowledged and the buffer is filled up to a certain degree of threshold, all the data accumulated in the output buffer are packed into one TCP segment and sent out at a time. After that, the output buffer, which is now empty, continues to buffer incoming data until the sent TCP segment is acknowledged and the buffer is filled up to a certain degree of threshold. The entire process iterates until the data transmission ends.

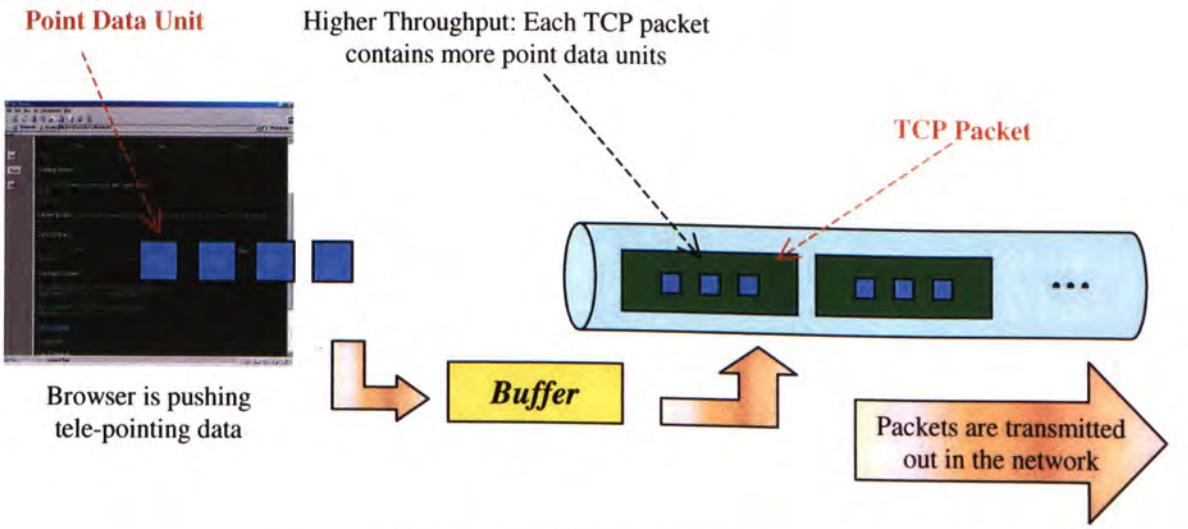


Figure 50 The Original Buffered Approach

According to the above methodology, the transmission throughput of each TCP packet is maximized. This is especially useful when the network bandwidth is scarce.

In contrast to the above buffered approach, sent-at-once approach sacrifices the original bandwidth conserving property but focuses on the instantaneous delivery. That is, once a data packet is generated on the sending side, it is sent immediately just after receiving the acknowledgement of the previous data packet. There is no accumulative data buffer for filling up each TCP segment before sending the data out. So, the throughput of each TCP packet, which refers to the percentage of the size of the data content over the total size of a TCP packet, is not maximized but the packet delay is minimized.

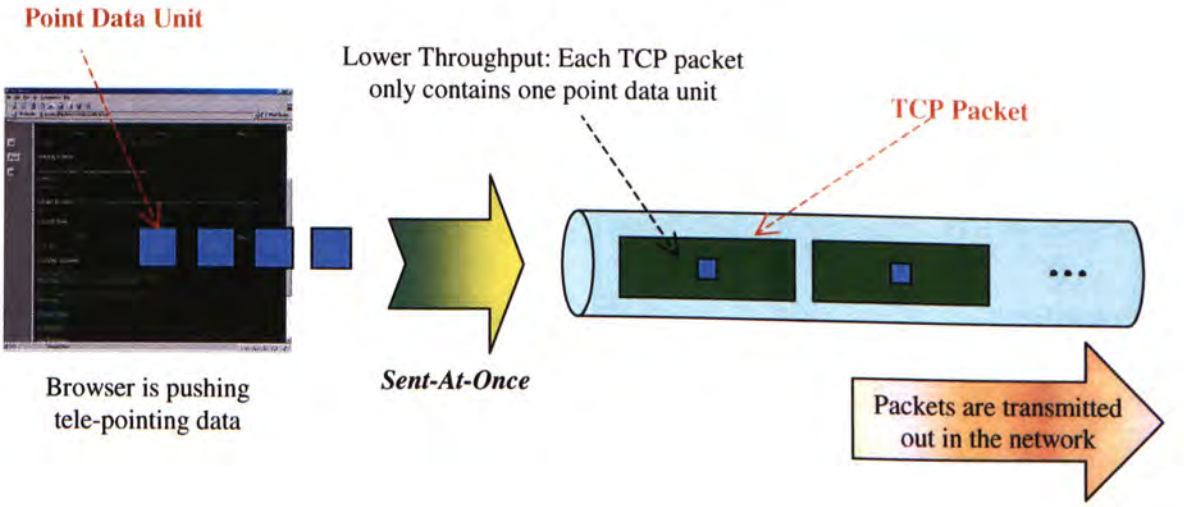


Figure 51 The Modified Sent-At-Once Approach

The tradeoff between the bandwidth consumption and the packet delay, affects the browser responsiveness. The sent-at-once approach is more suitable for

collaborative browsing, which emphasizes the real time browser responsiveness than the bandwidth consumption. This is because the feature synchronization of collaborative browsing does not consume a lot of bandwidth comparing with the bitmap screen transfer. Bandwidth requirement is not the bottleneck for collaborative browsing. Therefore, the sent-at-once approach is used in collaborative browsing.

5.2. Browser Responsiveness Study

5.2.1. Experiment Details

Instead of simple point-to-point communication, the collaborative browsing system is in a three-tier communication architecture, where all of the shared data have to pass from the source browser, then to the mediator and finally to the destination browser. Thus, data synchronization delay is a matter of concern. Larger delay can lead to lower operating responsiveness in the collaborative browser.

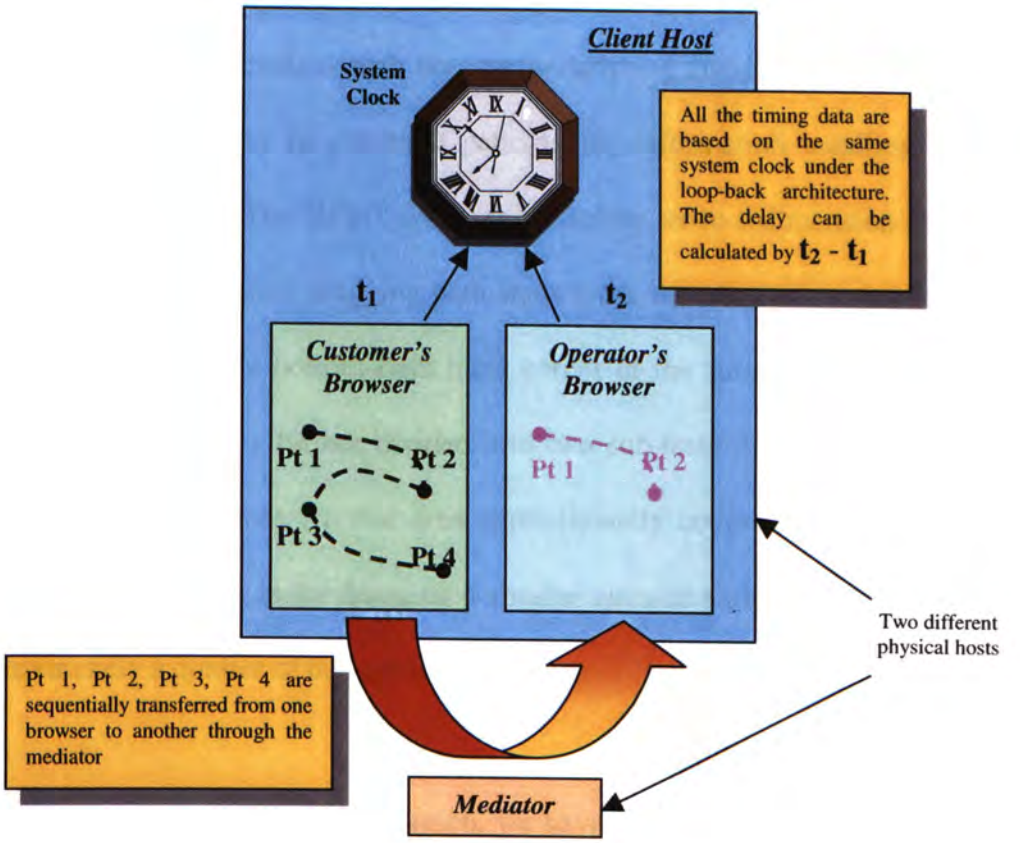


Figure 52 Loop-Back Tele-pointing Test

A loop-back test on the browser responsiveness was conducted to check if tele-pointing, the most demanding user interaction process, can be achieved in a reasonably fast time. The experiment test-bed is made up of one client host and one web server for testing. The equipment consists of two collaborative browsers of the same session run on the same computer and a mediator server running on another computer. The test starts by firstly synchronizing the web page contents of the two collaborative browsers. After that, we proceed to start the tele-pointing test. We deliberately move the mouse pointer in one collaborative browser so as to generate a sequence of tele-pointer movements in the other collaborative browser.

To generate the moving patterns for testing, we move the mouse pointer in a straight line path and a circular path because underlining and circling some texts or graphics are supposed to be common patterns for information handling in tele-pointing or whiteboard. The HTML frame is arbitrarily set to a dimension of 551 by 643 pixels. The straight line dragging path starts from the pixel of the top-left hand corner to the pixel of the bottom-right hand corner of the same HTML frame. The circular movement test is further divided into two sub-tests. One is for dragging a larger circular path, whose circular area approximately covers the full size of the HTML frame. The other is for dragging a smaller circular path, whose circular area is approximately a quarter of the whole frame size.

Moreover, for each type of motion, we have deliberately run the tests in two dragging speeds so as to identify any relationship between the latency and the dragging speed. The entire test is re-run after incorporating the packet delivery optimization technique described previously for comparison.

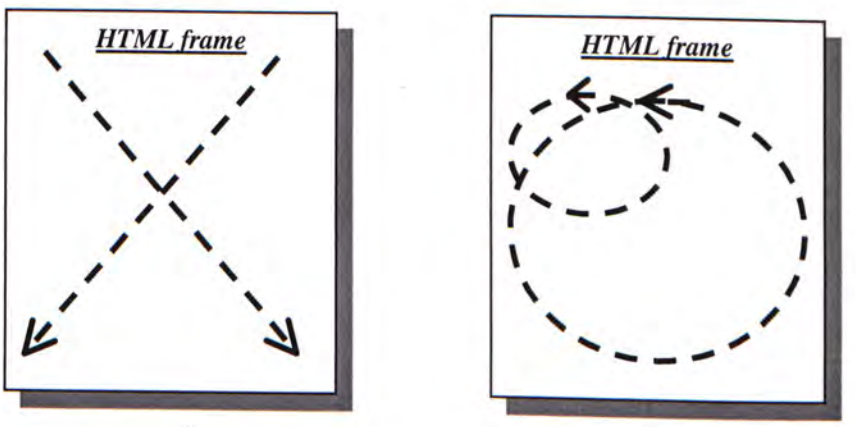


Figure 53 Dragging Patterns for the Loop-Back Test

During the mouse dragging, the time that the local mouse pointer coordinates are sampled at the active side before transmission and the time that the tele-pointer is generated at the passive side after receiving the tele-pointer geometric information are recorded in the system. Thus, the delay of synchronizing each of the tele-pointer coordinates between both sides can be obtained by doing a simple subtraction of the two timing data.

This configuration has the advantage of consistent timing as all the timing information is obtained based on the system clock of the same host. Therefore, the experiment is free of any distributed clock synchronization error [50].

5.2.2. The Assumptions

Some assumptions are made in this experiment:

1) For simplicity, we suppose there is no intermediate node in between the mediator in the web server and the testing host. Thus, the network architecture in the loop-back test is a direct browser-mediator-browser connection that involves only three network nodes. This is likely to be feasible in LAN or simple WAN environments.

2) The two collaborative browsers run on the same computer. But, in reality, the two collaborative browsers run on separate computers respectively. Thus, under this assumption, the processor workload of the computer is increased. Then the overall system performance is degraded and so, the accuracy of delay timing is deviated. However, such performance overhead is assumed to be negligible.

5.2.3. Experiment Results and Analysis

In the following graphs, the x-axis refers to the sequential numbering of each tele-pointer data unit sequentially sent from one browser to another browser while the y-axis refers to the round trip transmission time ($\mathbf{t_2}$ - $\mathbf{t_1}$) of each tele-pointer data unit. There are two mouse dragging speeds: Fast and Slow. For the straight line tests, "Fast" means the time period of dragging a straight line path once is one second while "Slow" means the time period is five seconds. For the circular dragging tests, "Fast" means dragging the quarter sized and full sized circular path ten times in five seconds and ten times in ten seconds respectively. "Slow" means dragging the quarter sized circular path five times in five seconds and the full sized one five times in ten seconds.

A) Under the Buffered Approach

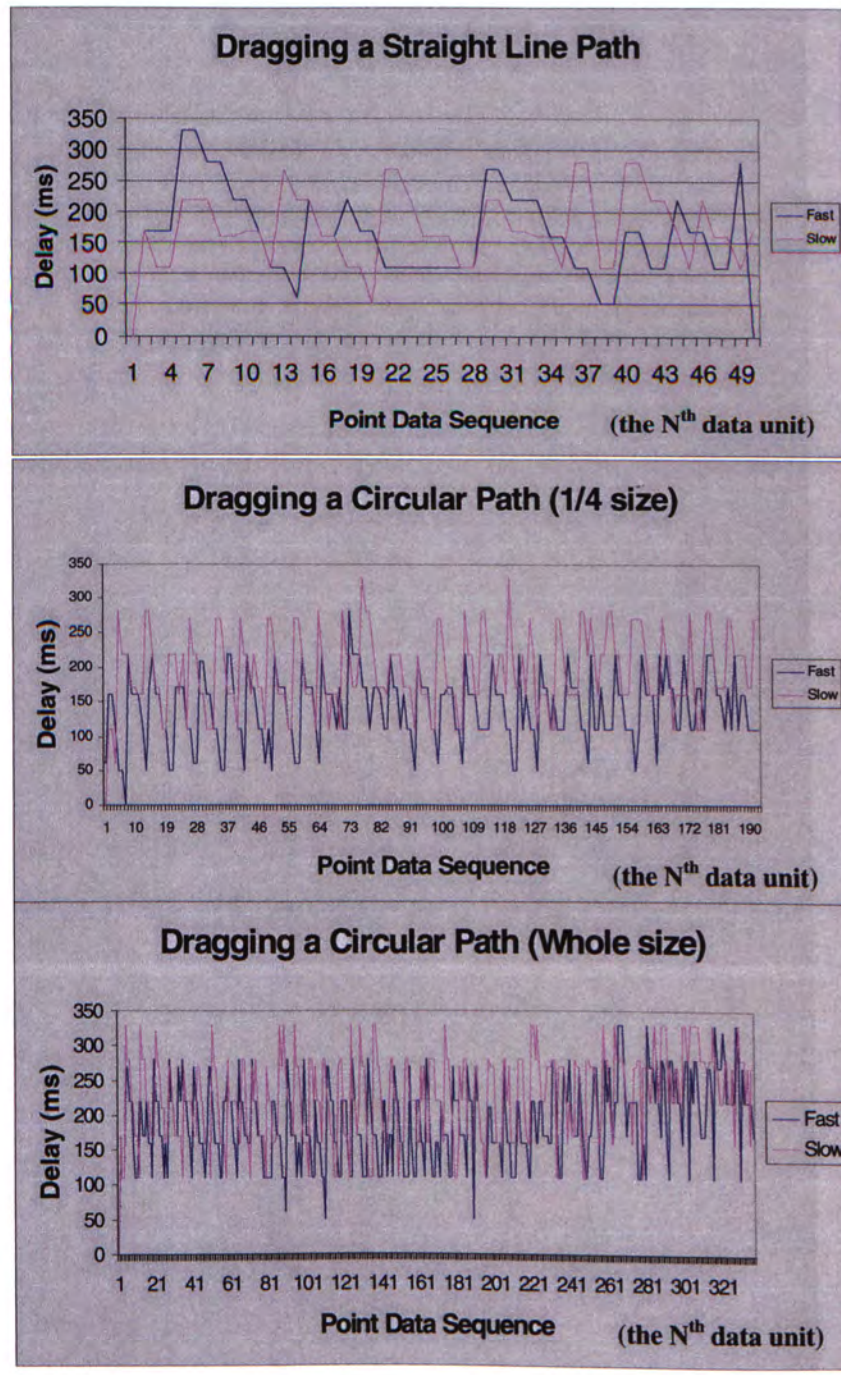


Figure 54 The Experimental Results for the Original Buffered Approach

The maximum delay of each tele-pointer data unit is no greater than 350 ms, regardless of the dragging speed and the dragging motion, and the average packet delay is about 175 ms.

B) The Sent-At-Once Approach

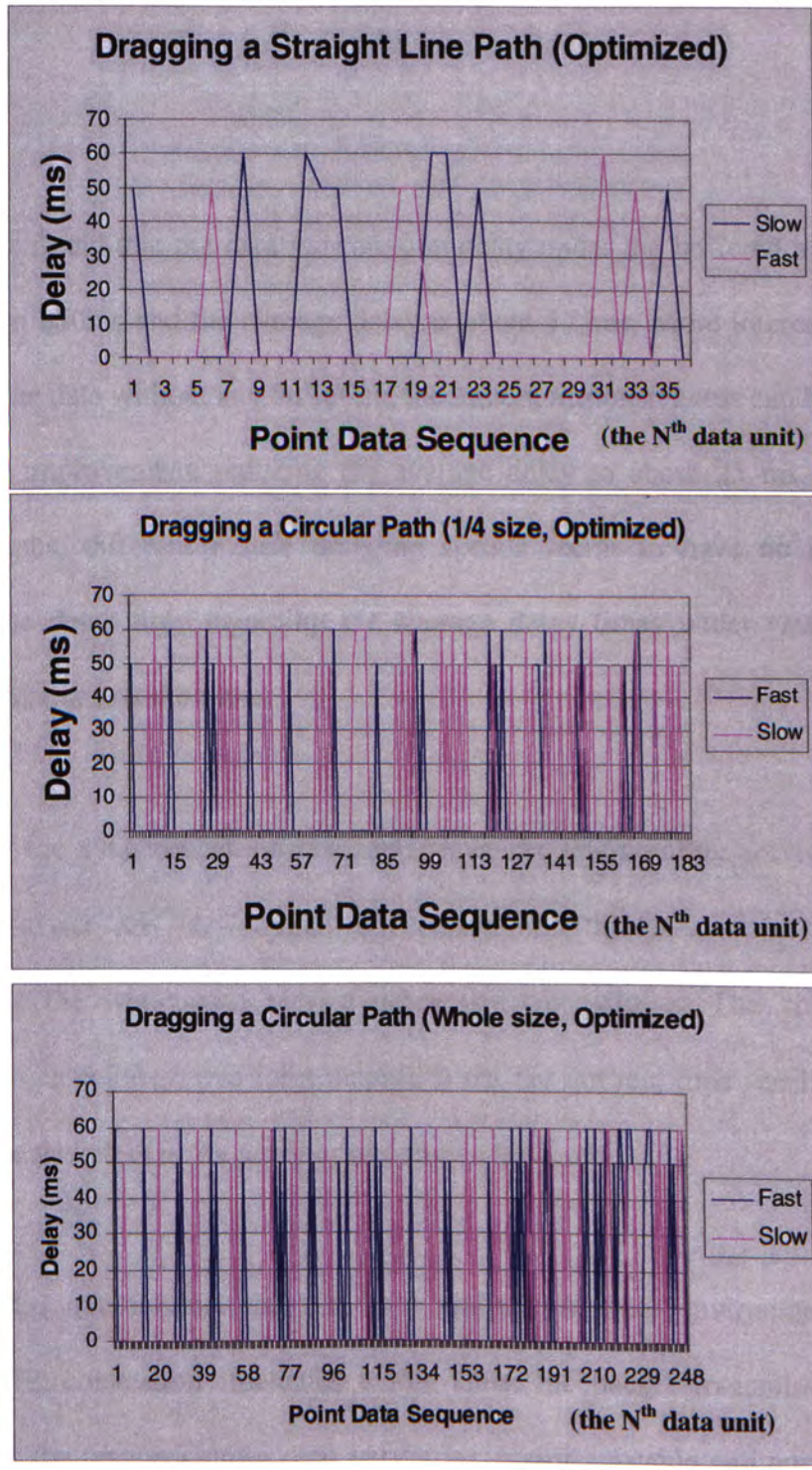


Figure 55 The Experimental Results for the Modified Sent-At-Once Approach

From the above graphs, the average delay is about 25 ms, which is a great improvement using the modified packet delivery approach.

C) Analysis

It was found that the data transmission delay under the buffered approach is no larger than 350ms and the average delay is about 175ms. More interestingly, by sending out the data without any buffering, the system responsiveness can have about a seven-fold improvement, reducing the average delay to about 25 ms. From the previous graphs, different mouse dragging speeds seems to have no significant impact on the delay time regarding the average delay times under two dragging speeds are more or less the same.

Since the synchronization mechanisms of the tele-pointing service and the whiteboard service are very similar as explained in the previous section, it is expected that the whiteboard service delay is to be similar. The collaborative services, such as collaborative form manipulation, are not real time sensitive. Thus, browser responsiveness is not a matter of concern for them.

We did not conduct the test in a realistic Internet environment, whose network traffic congestion fluctuates a lot. Thus, the packet transmission delay, which affects the responsiveness, can vary a lot, giving unstable and unreliable test results. However, the responsiveness performance under a LAN environment is supposed to be more or less close to the result of this loop-back test as there are fewer intermediate network nodes in a LAN.

5.3. Bandwidth Consumption

Based on the same test-bed as the loop-back tele-pointing test, a mouse pointer in the pushing side browser was dragged continuously, without any halt, to generate a worst-case point data pattern, which consumes the most bandwidth. The corresponding outgoing and incoming data rates are also recorded so that we can approximate the bandwidth consumption preliminarily. It should be noted that the incoming data rate and the outgoing data rate are symmetric because the data stream between the two collaborative browsers loops back into the same host.

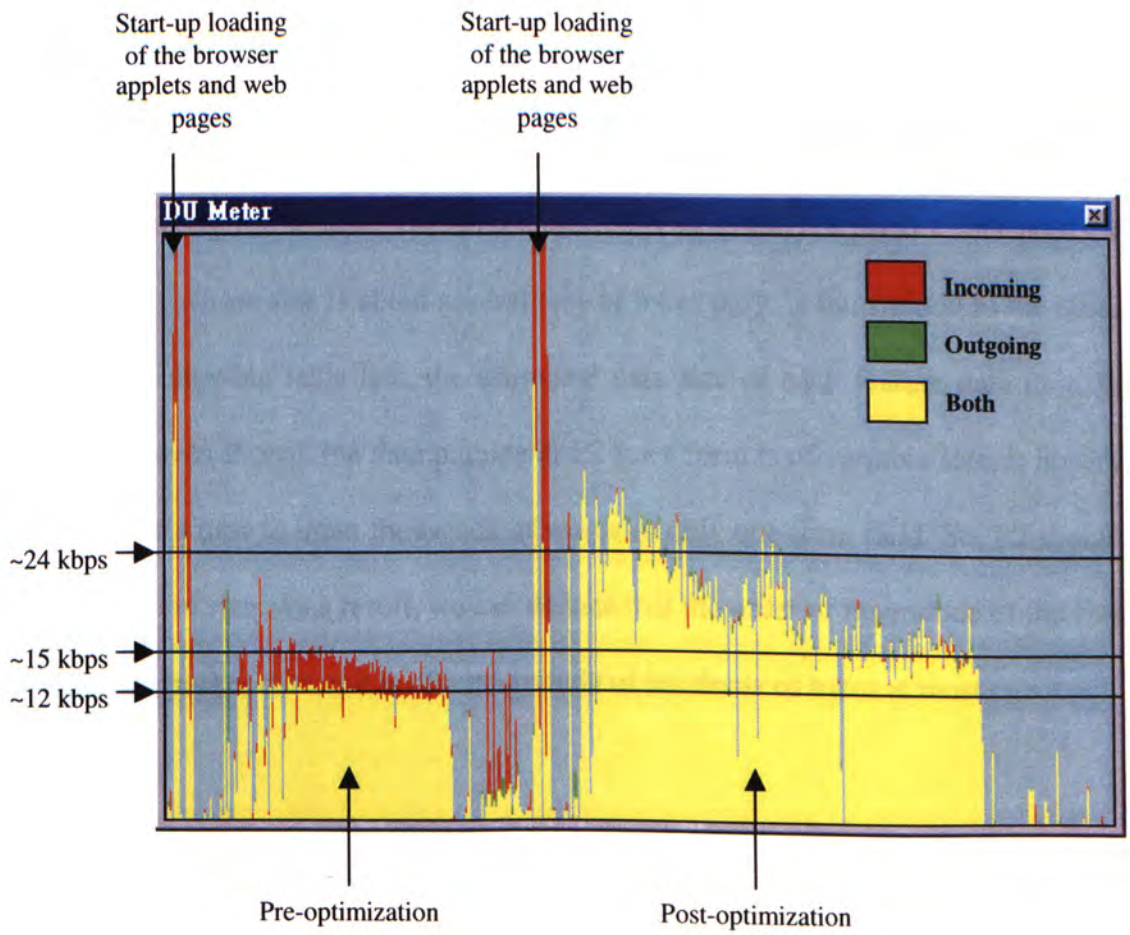


Figure 56 Bandwidth Consumption Before and After Optimization

The bandwidth consumption under the original buffered approach was found to be about 12 kbps. Under the sent-at-once approach, the bandwidth consumption was about 17 kbps.

From the result, it can be observed that the bandwidth consumption is so low that even a moderate speed modem can afford the above traffic. This efficiency cannot be achieved by the bitmap screen transfer approach, which involves the transfer of megabits of screen bitmaps.

The low bandwidth consumption of our system can be attributed to the small packet size under the feature synchronization approach. For example, using the bitmap transfer approach for tele-pointing service, a simple mouse move on one side triggers the transmission of a series of bitmap screen images to the other side. However, using the feature synchronization approach, only a packet data unit for tele-pointer, whose size is about several tens of bytes only, is transmitted to the other side. The following table lists the estimated data size of each feature data unit. In particular, even though the data parameter P2 for a form is of variable size, it is very unlikely for a user to input thousands of words in only one form field. So, P2 should not be large in size. As a result, we can deduce that the order of magnitude of the size of a typical packet data unit is of a magnitude of hundreds of bytes at most.

Feature	Estimated Data Size		
	CMD	Data Parameter 1 (P1)	Data Parameter 2 (P2)
Control Access	access	1 integer	-
Tele-pointer	ptr	1 integer (x-coordinate of a tele-pointer)	1 integer (y-coordinate of a tele-pointer)
Whiteboard	wb	1 integer (x-coordinate of a whiteboard point data)	1 integer (y-coordinate of a whiteboard point data)
Clear Form	clrfrm	-	-
Form Data	form	2 integers for addressing a form field	Text, Numbers or Boolean values (form field value)
Web Page Push	String representing a URL	1 integer (position of the horizontal scrollbar)	1 integer (position of the vertical scrollbar)

Table 3 Estimated Data Size of the Packet Data Unit for each Feature Type

6. Conclusions

This thesis focuses on the engineering experience of realizing a collaborative browsing system specially designed for a digital call center. Collaborative features are incorporated into an object-based web document model so that we eventually come up with a Shareable Document Object Architecture (SDOA). The proposed three-tier communication architecture further enhances the system functionality by using the mediator to process various intermediate data for Hinting and collaborative IVR service.

Using the above architecture as the basis, basic collaborative services, such as tele-pointing, shared whiteboard, web page pushing, become feasible under the feature synchronization approach. Experiments showed that the bandwidth consumption can be as low as about 17 kbps, making it feasible to use our system through a dialup modem connection as well as a WAN or a LAN. We have also found that the sent-at-once packet transmission methodology can lead to very fast system response time. The average packet delay time is seven times faster than the buffered packet transmission methodology. This is crucial to the real time sensitive collaborative services, like tele-pointing, shared whiteboard.

Still, our work has are some limitations. The shared contents for collaboration are confined to the standardized HTML documents, like HTML web pages, HTML form. However, there are some proprietary web documents that are not based on the HTML format. For example, some web-based form documents [51], [52] require legacy applications for inputting the form data, instead of using a web browser. As a

result, the collaborative features of our system are not applicable to these documents. But, knowing that the popularity of using the standardized HTML documents is higher than that of using the proprietary web documents, our system can be applied in most cases. Currently, our system only supports the HTML format, but not the XML format, which will be considered for further development.

Furthermore, using collaborative browsing simply increases the average time spent by an operator for a customer call. This is because the average interaction time between an operator and a customer increases when using collaborative browsing as well as the original voice communication. As the time spent by an operator for a customer is a critical cost factor in the deployment of a call center, collaborative browsing, which is very user friendly, may not be cost-effective and suitable for call center service in reality.

As a whole, our first major achievement lies on smoothing out the web form handling difficulty through collaboration. Instead of simple form data synchronization between a pair of collaborative browsers, collaboration is further enhanced by Hinting, which relates the form fields to the backend database to show the hidden data relationships. These data relationships are beneficial for operators to interpret the semantics of the related form fields so as to help a troubled customer to fill in a complicated web form. With Hinting, an operator can collaboratively fill in a web form with a customer too.

The second major achievement is the Phone-to-PC collaborative Interactive Voice Response (IVR), which is a variant of the PC-to-PC collaborative browser. By

reducing the traditional IVR user interaction methodology to a form-based mechanism, the concept of Hinting is re-used here to support collaboration. This revolutionizes the traditional customer self-served IVR service to an operator assisted collaborative IVR service. Firstly, Telephone Forms are constructed for the IVR services. Secondly, the corresponding IVR service tree serves as hints for an operator to view the traversal status of a customer. With collaborative IVR, an operator can assist a customer in accessing different IVR services under a complicated and highly customized IVR service tree in a quick and efficient manner.

Most of the existing systems, such as the MS Netmeeting which targets on the generic collaborative computing, are based on PC-to-PC data collaboration through proprietary applications. On the contrary, our research focus aims at paving ways to improve collaborative efficiency through a content-driven methodology rather than the application sharing methodology. Handling the complexity of web forms using hints during collaboration is also a core feature of our work. Also, when compared with some of the existing commercial or academic systems [53], [54], [55], our system has the unique feature to support collaborative IVR.

Collaborative browsing is one of the three parts in the construction of the full-featured digital call center. For a complete prototype, we will have to further integrate with the ACD and the VoIP communication system. After system integration, rigorous on-site tests will be conducted for further enhancement.

Also, in the future research, form data protection for customer privacy should be supported in such way that a customer can selectively filter some private form

data, such as home address, during collaboration. Also, phone-based data input method specially for Telephone Form is needed so that a customer can enter some more complicated data, such as alphanumeric strings, into a Telephone Form via a phone.

To conclude, we have achieved a web-based collaborative browsing system for a digital center. We have shown the ways to revolutionize and enhance the traditional voice-only customer support service in call centers.

Appendix A — Government Profit Tax Return Form

The following is a real government tax return form [56] in Hong Kong. The details of how to fill in the boxes in section 2.2 of this form are shown on the next pages.

INLAND REVENUE DEPARTMENT
PROFITS TAX RETURN—PERSONS OTHER THAN CORPORATIONS
FINAL ASSESSMENT
AND PROVISIONAL PAYMENT

IN ANY COMMUNICATION PLEASE QUOTE THE FILE NUMBER BELOW

FILE NO.
To

Revenue Tower,
5 Gloucester Road,
Wan Chai, Hong Kong.
G.P.O. Box 132, Hong Kong.
Web site:
http://www.info.gov.hk/ird
Tel. No.:

SAMPLE

By virtue of Section 51(1) of the Inland Revenue Ordinance (Cap. 112), you are required to make on this form a true and correct return of the Assessable Profits (or Adjusted Loss) (See Note 1) arising during the basis period (See Note 2) for the year of assessment ended 31 March .

All sections/boxes of the form **MUST** be completed and returned to me **WITHIN 1 MONTH** from the date of this Notice. Return by facsimile is not acceptable. You should read the attached Notes and Instructions before completing the form.

If you are **NOT** a **SMALL** business (See Note 3), you **MUST** submit, together with this form, the following documents (collectively called "Supporting Documents"):

(a) a certified copy of your Balance Sheet and Profit and Loss Account in respect of the basis period;
(b) a tax computation with supporting schedules showing how the amount of Assessable Profits (or Adjusted Loss) has been arrived at; and
(c) other documents as specified below and in the attached Notes and Instructions.

If you are a **SMALL** business, you need not submit the Supporting Documents with this form. However, you **MUST**:

(a) prepare the Supporting Documents before you complete this form;
(b) complete this form in accordance with the Supporting Documents; and
(c) retain the Supporting Documents as you may be required to submit them later.

Date:

Assistant Commissioner

If space is insufficient, provide particulars on a separate sheet — **Exclude cents when stating amounts.**

1.	**STATEMENT OF ASSESSABLE PROFITS OR ADJUSTED LOSS (See Note 4)** *(To facilitate the making of adjustments to arrive at the assessable profits or adjusted loss, a pro forma tax computation form is attached.)*		
1.1	Assessable Profits (before loss brought forward):	If NIL, enter "0"	HK$
1.2	Adjusted Loss (before loss brought forward):	If NIL, enter "0"	HK$

2. PERSONAL PARTICULARS OF PROPRIETOR OR PARTNER AND ALLOCATION OF PROFITS / (LOSS) (See Note 5)

2.1 PERSONAL PARTICULARS OF PROPRIETOR OR PARTNER *(Go to box 2.2)*

	Full Name	Residential address	ONLY for partner(s) who entered/left during the basis period: Date entered	Date left
(1)				
(2)				
(3)				
(4)				

This part involves complicated set of data dependencies. Please refer to the next page for the steps to fill in these boxes.

2.2 ALLOCATION OF ASSESSABLE PROFITS (OR ADJUSTED LOSS)

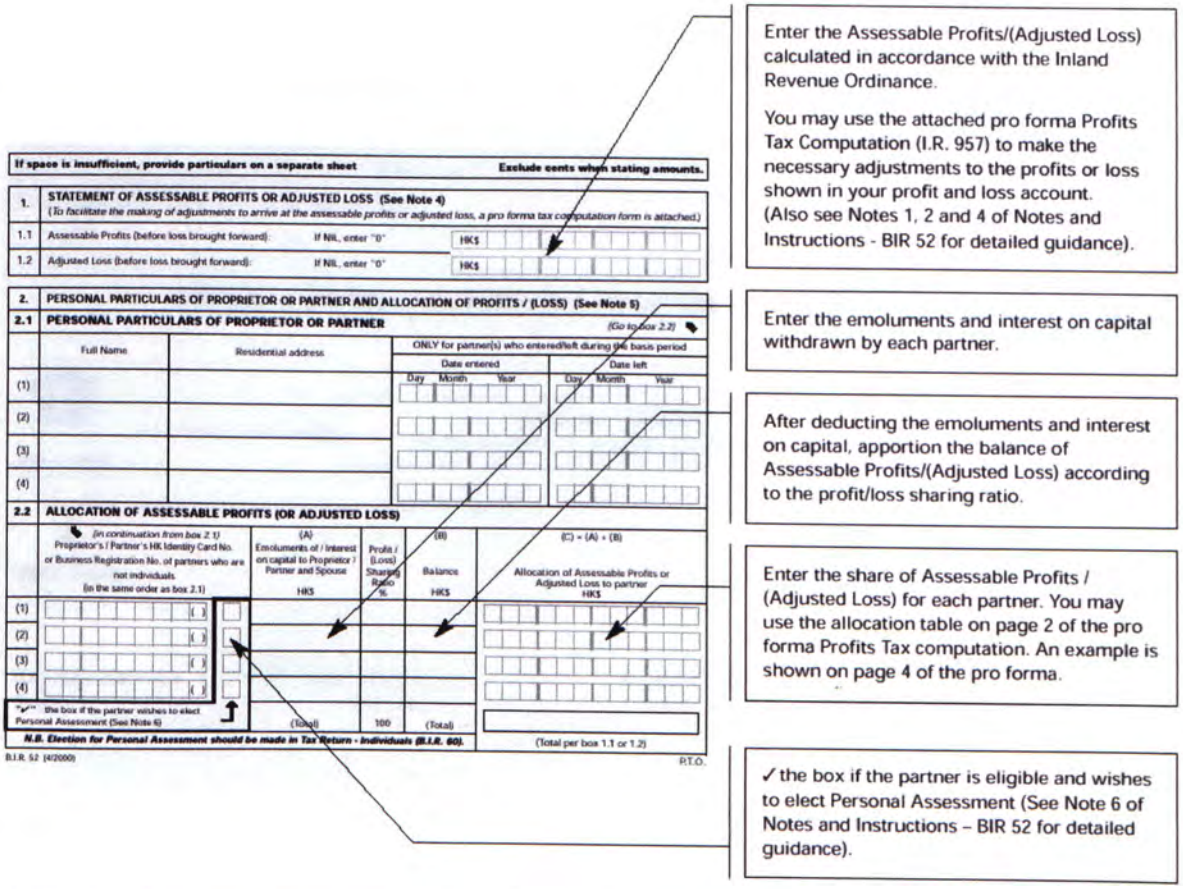
	(in continuation from box 2.1) Proprietor's / Partner's HK Identity Card No. or Business Registration No. of partners who are not individuals (in the same order as box 2.1)	(A) Emoluments of / Interest on capital to Proprietor / Partner and Spouse HK$	Profit / (Loss) Sharing Ratio %	(B) Balance HK$	(C) = (A) + (B) Allocation of Assessable Profits or Adjusted Loss to partner HK$
(1)	( )				
(2)	( )				
(3)	( )				
(4)	( )				
	(Total)		100	(Total)	(Total per box 1.1 or 1.2)

"✓" the box if the partner wishes to elect Personal Assessment (See Note 6)

N.B. Election for Personal Assessment should be made in Tax Return - Individuals (B.I.R. 60).

B.I.R. 92 (4/2008)

P.T.O.



If space is insufficient, provide particulars on a separate sheet — Exclude cents when stating amounts.

1.	STATEMENT OF ASSESSABLE PROFITS OR ADJUSTED LOSS (See Note 4) (To facilitate the making of adjustments to arrive at the assessable profits or adjusted loss, a pro forma tax computation form is attached.)		
1.1	Assessable Profits (before loss brought forward):	If NIL, enter "0"	HK$
1.2	Adjusted Loss (before loss brought forward):	If NIL, enter "0"	HK$

2.	PERSONAL PARTICULARS OF PROPRIETOR OR PARTNER AND ALLOCATION OF PROFITS / (LOSS) (See Note 5)					
2.1	PERSONAL PARTICULARS OF PROPRIETOR OR PARTNER (Go to box 2.2)					
	Full Name	Residential address	ONLY for partner(s) who entered/left during the basis period: Date entered (Day Month Year)		Date left (Day Month Year)	
(1)						
(2)						
(3)						
(4)						

2.2	ALLOCATION OF ASSESSABLE PROFITS (OR ADJUSTED LOSS)				
	(In continuation from box 2.1) Proprietor's / Partner's HK Identity Card No. or Business Registration No. of partners who are not individuals (in the same order as box 2.1)	(A) Emoluments of / Interest on capital to Proprietor / Partner and Spouse HK$	Profit / (Loss) Sharing Ratio %	(B) Balance HK$	(C) = (A) + (B) Allocation of Assessable Profits or Adjusted Loss to partner HK$
(1)	( )				
(2)	( )				
(3)	( )				
(4)	( )				
	"✓" the box if the partner wishes to elect Personal Assessment (See Note 6)	(Total)	100	(Total)	(Total per box 1.1 or 1.2)

N.B. Election for Personal Assessment should be made in Tax Return - Individuals (B.I.R. 60).

B.I.R. 52 (4/2000) — P.T.O.

Enter the Assessable Profits/(Adjusted Loss) calculated in accordance with the Inland Revenue Ordinance.

You may use the attached pro forma Profits Tax Computation (I.R. 957) to make the necessary adjustments to the profits or loss shown in your profit and loss account. (Also see Notes 1, 2 and 4 of Notes and Instructions - BIR 52 for detailed guidance).

Enter the emoluments and interest on capital withdrawn by each partner.

After deducting the emoluments and interest on capital, apportion the balance of Assessable Profits/(Adjusted Loss) according to the profit/loss sharing ratio.

Enter the share of Assessable Profits / (Adjusted Loss) for each partner. You may use the allocation table on page 2 of the pro forma Profits Tax computation. An example is shown on page 4 of the pro forma.

✓ the box if the partner is eligible and wishes to elect Personal Assessment (See Note 6 of Notes and Instructions – BIR 52 for detailed guidance).

B-4 Allocation of Assessable Profits or Adjusted Loss (item 29 of pro forma computation)

1. General
   - Assessable Profits (or Adjusted Loss) will be allocated to each partner simply according to his/her profit and loss sharing ratio when none of the partners have drawn emoluments or interest on capital (which is nothing more than a fixed share of profits).
   - If partners have drawn emoluments or interest on capital, these amounts will be taken into account in the allocation before the residue is shared among the partners.
   - In no case will a partner be allocated a share of loss if the partnership has an overall Assessable Profit. Similarly, a partner will not be allocated a share of profit if the partnership has an overall Adjusted Loss.

2. Example of Allocation of Assessable profits (or Adjusted loss) among partners

The partners of a business Mr CHAN, Mr CHOI & Ms LI drew salary of $320,000, $200,000 and $50,000 respectively. After adding back their salaries which are disallowable for tax purposes, the partnership's assessable profits were $210,000. The partners shared profits or loss equally.

Hidden fields for intermediate calculations

(A) Partners	(B) Emoluments and interest on capital	(C) Profit/Loss Sharing Ratio	(D) Share of Balance	(E) Sub-total	(F) Re-allocation	(G) Share of Assessable Profits or Adjusted Loss
	$		$	$	$	$
Mr CHAN (1)	320,000 (1)	1/3 (1)	−120,000 (4)	200,000 (5)	−50,000 (7)	150,000 (8)
Mr CHOI (1)	200,000 (1)	1/3 (1)	−120,000 (4)	80,000 (5)	−20,000 (6)	60,000 (8)
Ms LI (1)	50,000 (1)	1/3 (1)	−120,000 (4)	−70,000 (5)	+70,000 (7)	Nil (8)
Total	570,000 (1)		−360,000 (3)	210,000 (5)		210,000 (2)

(The numbers in brackets demonstrate the sequence of the steps in working through the calculations.)

Step (1) Fill in the names, emoluments/interest on capital drawn and profit/loss sharing ratios in the respective columns and add up the total of column (B).

Step (2) Fill in the Assessable Profits $210,000 at the total position of column (G)

Step (3) Compute the total of column (D) by subtracting the total of column (B) from the total of column (G). The resultant amount may be positive or negative. In this example, it is negative.
$210,000 − $570,000 = −$360,000

Step (4) Allocate the total balance in accordance with each partner's profit/loss sharing ratio and fill in column (D).
−$360,000 × 1/3 = −$120,000 for each partner

Step (5) Add up the emoluments/interest on capital (B) and the share of balance (D) for each partner and fill in his/her respective share of Assessable Profits/Adjusted Loss at column (E).
Mr Chan : $320,000 − $120,000 = $200,000
Mr Choi : $200,000 − $120,000 = $80,000
Ms Li : $50,000 − $120,000 = −$70,000

Step (6) If all the amounts in column (E) are positive for a partnership having an overall assessable profit, or all the amounts are negative for a partnership incurring an overall adjusted loss, the allocation is complete. Each partner's share of assessable profits or adjusted loss is shown in column (E).

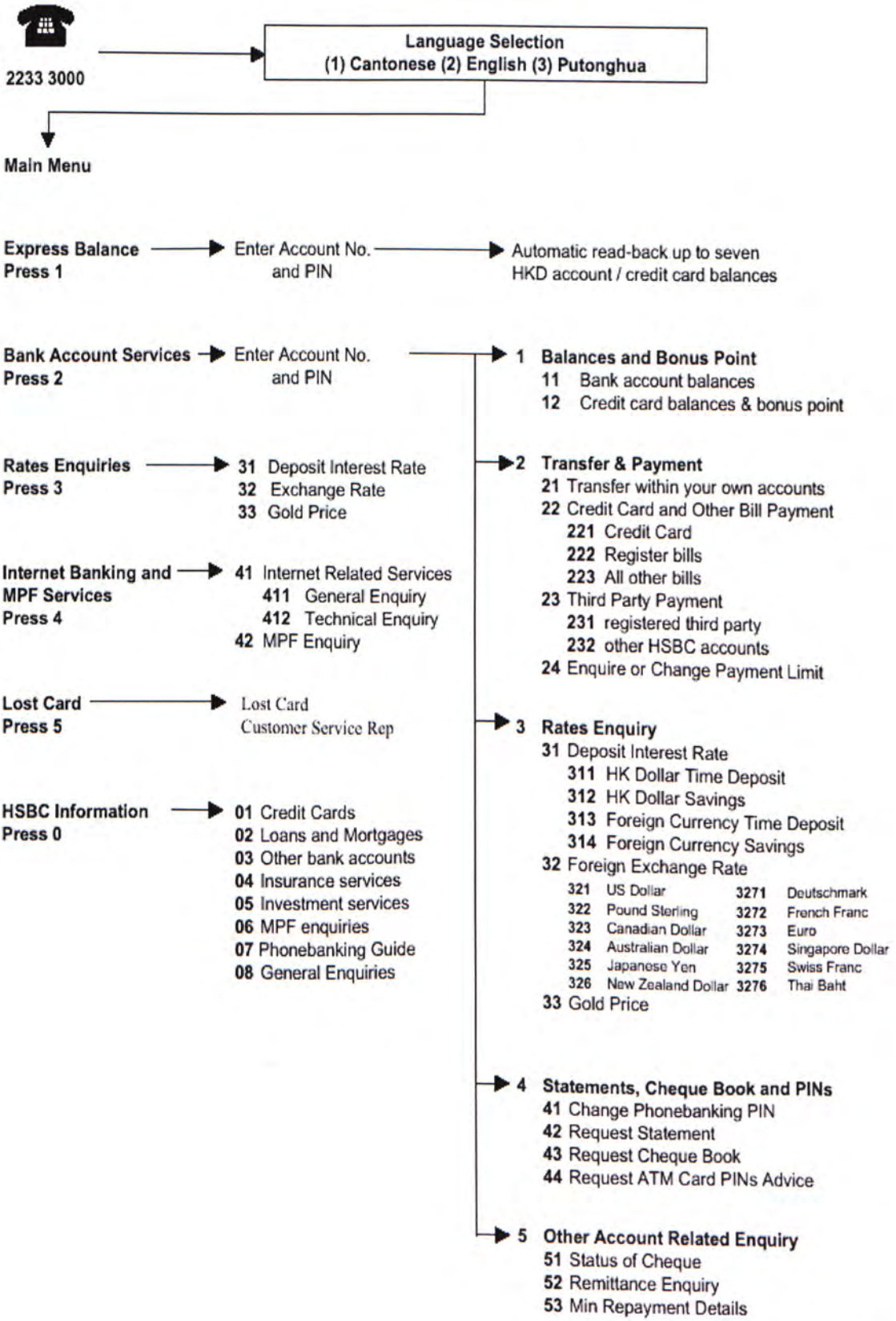
Step (7) However, a reallocation is necessary if there are both positive and negative amounts in column (E), as in this example. The negative amounts (in an overall assessable profit case) or the positive amounts (in an overall adjusted loss case), are to be reallocated to the other partners in proportion to their amounts in column (E). Insert the reallocated amounts in column (F). In this example, Ms Li's negative amount of $70,000 is reallocated in the ratio of 20 : 8 to Mr Chan and Mr Choi.
Mr Chan : $70,000 × $200,000/($200,000 + $80,000) = $50,000
Mr Choi : $70,000 × $80,000/($200,000 + $80,000) = $20,000

Step (8) After the reallocation, add up the amounts in columns (E) and (F) to arrive at the amounts shared by each partner in column (G). Check whether all the rows and columns can be reconciled. If so, the allocation is complete.

Appendix B — A Phone Banking IVR Service Tree

The following is a user guide of how to use the Interactive Voice Response service for phone banking [57] in Hongkong and Shanghai Banking Corporation (HSBC). This user guide logically depicts the IVR service tree for phone operation.

HSBC's 24-hour automated phonebanking service



Bibliography

[1] Customer Relationship Management Literature and Research, http://www.crm-forum.com/lire/ppr.htm

[2] J. Davidson, J. Peters, Voice over IP Fundamentals, P.40-42, Cisco Press, 2000.

[3] S. Dellutri, Beyond Middlesware: The Web-Enabled Call Center, http://www.tmcnet.com/tmcnet/articles/cosmo0499.htm

[4] K. Dawson, Call Center Savvy, Telecom Books, 1999.

[5] C. C. Leung, K. N. Yuen, H. C. Ho, Y. S. Moon, An IP-based Call Center with Fault Tolerance Design, *to be appeared in Proceedings of the 12th International Symposium on Software Reliability Engineering*, Hong Kong, Nov. 27-30, 2001.

[6] W. Yarberry, Computer telephony integration, CRC, 2000.

[7] Y. S. Moon, C. C. Leung, K. N. Yuen, H. C. Ho, X. Yu, A CRM model based on Voice over IP, *In Proceedings of the 2000 IEEE Canadian Conference on Electrical & Computer Engineering*, Canada, May 7-10, 2000.

[8] K. N. Yuen, H. C. Ho, C. C. Leung, Y. S. Moon, Voice over IP Application in Education – VoIP Enabled Web Lecture, *In Proceedings of the 4th Global Chinese Conference on Computers in Education*, Republic of Singapore, May 29-31, 2000.

[9] M. Kobayashi, M. Shinozaki, T. Sakairi, M. Touma, S. Daijavad, C. Wolf, Collaborative customer services using synchronous Web browser sharing, *In Proceedings of the ACM 1998 conference on Computer supported cooperative work*, P.99-108, Seattle, WA USA, Nov. 14 - 18, 1998.

[10] M. C. Dah, D. M. Griffin, Building Collaboration Software for the Internet, *Digital Technical Journal*, Vol. 8, No. 3, P.66-74, 1996.

[11] K. N. Yuen, A Comprehensive VoIP System with PSTN Connectivity, M. Phil. Thesis, Dept. of Computer Science & Engineering, The Chinese University of Hong Kong, June, 2001.

[12] C. C. Leung, An Intelligent IP-based Call Center with Fault tolerance Design, M.Phil. Thesis, Dept. of Computer Science & Engineering, The Chinese University of Hong Kong, June, 2001.

[13] G. Held, Voice Over Data Networks, McGraw-Hill, 1998.

[14] M. Goncalves, Voice Over IP Networks, P.103, McGrawHill, 1999.

[15] G. Sidler, A. Scott, H. Wolf, Collaborative Browsing in the World Wide Web, *In Proceedings of the 8th Joint European Networking Conference*, Edinburgh, May 12-15, 1997.

[16] M. Fuchs, Let's Talk: Extending the Web to Support Collaboration, *In Proceedings of WET ICE'96*, P.316-321, 1996.

[17] T. Chen, J. S. Sung, World Wide Meet (WWM) —Browsing the web not alone, *In Proceedings of the International Conference on Multimedia Telecommunication in Management*, Hong Kong Baptist University, Springer Verlag, Dec. 1998.

[18] T. A. Phelps, R. Wilensky, Multivalent Annotations, *In Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, Pisa, Italy, Sep. 1-3, 1997.

[19] Interactive Government Services Directory, http://www.igsd.gov.hk.

[20] Payment-by-Phone Service, https://www.ppshk.com.

[21] S.F. Li, A. Hopper, A Framework to Integrate Synchronous and Asynchronous Collaboration, *In Proceedings of the Seventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1998. (WET ICE '98).

[22] T. Sakairi, M. Shinozaki, M. Kobayashi, CollaborationFramework: a toolkit for sharing existing single-user applications without modification, *In Proceedings of the 3rd Asia Pacific Conference on Computer Human Interaction*, 1998.

[23] S. Gianoutsos, W4: A World Wide Web Browser with CSCW support, *In Proceedings of the Sixth Australian Conference on Computer-Human Interaction*, 1996.

[24] F. Sadri, Data dependencies in the relational model of databases: a generalization, Thesis (Ph.D.) of Princeton University, 1980.

[25] H. Robinson, Facts, dependencies and data models, Milton Keynes: Open University, 1990.

[26] Xform Requirements, http://www.w3.org/TR/2000/WD-xhtml-forms-req-20000821.

[27] Schema Design Concepts, http://www.dmtf.org/educ/tutorials/cim/extend/design.html.

[28] P. Litwin, Fundamentals of Relational Database Design, http://www.microsoft.com/TechNet/Access/technote/ac101.asp, 2000.

[29] J. Bowers, D. Martin, Machinery in the new factories: interaction and technology in a bank's telephone call centre, *In Proceedings on the ACM 2000 Conference on Computer supported cooperative work*, P.49-58, Philadelphia, PA USA, Dec. 2 - 6, 2000.

[30] C. Porciuncula, Conceptual Graph Representation of Interactive Voice Response (IVR) Applications, *Part of the RMIT CS433 Knowledge Representation Course Assessment*, 1995

[31] Hypertext Transfer Protocol 1.1, Section 1.4, http://www.ietf.org/rfc/rfc2068.txt.

[32] D. Atkins, T. Ball, M. Benedikt, G. Bruns, K. Cox, P. Mataga, K. Rehor, Experience with a domain specific language for form-based services, *In Usenix Conference on Domain Specific Languages*, Santa Barbara, CA, October 1997.

[33] D. Atkins, Integrated Web and telephone service creation, *Bell Labs Technical Journal*, 2(1): 19-35, 1997.

[34] P. Resnick, Phone-Based CSCW: Tools and Trials, *ACM Transactions on Information Systems*, Vol. 11, No. 4, P.401-424, 1993.

[35] Douglas E. Comer, Internetworking with TCP/IP Volume I, Prentice Hall, 1999.

[36] Java RMI, http://java.sun.com/products/jdk/rmi/index.html.

[37] A. S. Tanenbaum, Computer Networks, P.36-37, P.521-545, Prentice Hall, 1996.

[38] V. Jagannathan, G. Almasi, A. Suvaiala, Collaborative infrastructures using the WWW and CORBA-based environments, *In Proceedings of the 5th Workshop on the Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1996.

[39] Signed Java Applet, http://java.sun.com/sfaq.

[40] Netscape Plug-in Development, http://developer.netscape.com/docs/manuals/communicator/plugin/index.htm.

[41] Netscape Navigator Object Hierarchy http://developer.netscape.com/docs/manuals/communicator/jsguide4/navobj.htm

[42] Netscape Communicator, http://www.netscape.com

[43] A. Umar, Object-oriented client/server Internet environments, Prentice Hall, 1997.

[44] LiveConnect, http://home.netscape.com/eng/mozilla/3.0/handbook/plugins/index.html.

[45] PCI analog telephony interface card used in our project, http://www.nmss.com/nms/nmssweb.nsf/productlist/AG_2000_Series.

[46] P. Resnick, R. A. Virzi, Relief from the Audio Interface Blues: Expanding the Spectrum of Menu, List, and Form Styles, *CCS (Centre for Coordination Science) Working Paper No. 184*, P.3633-93, 1995.

[47] T. Hornstein, Telephone Voice Interfaces on the Cheap, *In Computer Science Research at UBILAB*, P.134--146, Zurich, 1994.

[48] M. Detweiler, R. JR. Schumacher, N. JR. Gattuso, Alphabetic input on a telephone keypad, *In Proceedings of the 34th Annual Meeting of the Human Factors Society*, Santa Monica, Calif., 1990.

[49] L. Fast, R. Ballantine, Dialing a name: Alphabetic entry through a telephone keypad, *SIGCHI Bull. 20*, 2, 34, 1988.

[50] G. Coulours, J. Dollimore, T. Kindberg, Distributed Systems Concepts and Design, P.287-310, Addison Wesley, 1999.

[51] R. Bentley, W. Appelt, Designing a system for cooperative work on the World-Wide Web: experiences with the BSCW system, 1997, *In Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, Vol. 4, P.297-306, Jan. 7-10, 1997.

[52] O. Marques, S. Hsu, Assessing the Feasibility of Using Microsoft NetMeeing in Distance Education, *In Proceedings of ICECE'99*, P.109-113, Rio de Janeiro, Brazil, Aug. 11-14, 1999.

[53] S. Jacobs, M. Gebhardt, S. Kethers, W. Rzasa, Filling HTML Forms Simultaneously: CoWeb – Architecture and Functionality, *In Proceedings of the Fifth International World Wide Web Conference*, P.1385-1395, May, 1996.

[54] Electronic Form System by the Hong Kong Government, http://www.igsd.gov.hk/eform.

[55] EZ-Form, http://www.ezx.com/faq.htm#eform

[56] Details of Profit Tax Return in Hong Kong, http://www.info.gov.hk/ird/h_compf.htm.

[57] HSBC Phone Banking Services, http://www.hsbc.com.hk/hk/personal/bank/phone.htm.