AUTOMATED REMOTE DATABASE RETRIEVAL THROUGH TELEPHONE.

BY
Athar Mohsin Zaidi

A DISSERTATION

Submitted to

Faculty of National University of Science and ~~technology~~ Technology (NUST)

In partial fulfillment of the requirements

For the degree of

MASTERS OF COMPUTER SCIENCES

Department of Computer Sciences

2000

ABSTRACT

AUTOMATED REMOTE DATABASE RETRIEVAL THROUGH TELEPHONE.

By

Athar Mohsin Zaidi

Base workshops of Corp of E.M.E play an important role in managing requisite level of vehicles / equipment availability to the field units of Pakistan Army. These includes central, base, combined and area workshops. Vehicle / equipment, which is beyond the repair capability of field workshops, is back loaded to ordnance depots and then fed to the base workshops concern for necessary overhaul. Subsequently overhauled vehicle / equipment is returned to ordnance depots for their subsequent issue to the units. In addition to this channel the vehicles / equipment are entertained by the base workshops directly from the units after obtaining necessary approval from their respective Commander EME and EME Directorate (G.H.Q). Units who wish to utilize expertise of base workshop faces a lengthy correspondence before and after sending their vehicles / equipment, to keep track of repair of their vehicles / equipment. At the same time base workshops also gets engaged in correspondence with the units to inform them about the progress of repair. Before sending the vehicles to base workshops, Field units route their repair requests to GHQ through CEME. GHQ then asks the feasibility report from Base workshop, in response Base workshop sends a deposit date for repair.

To avoid lengthy correspondence there is a need to establish a dedicated system at base workshop that provide current information to the units regarding availability of repair space in workshop as well as the repair status of their vehicles / equipment, at any time they need to know. This could be achieved if field units get the required information from base workshop on telephone. With the provision that the in coming calls be attended by the computer. The proposed project will be focussed on establishing such an atomized system. The basic idea of the project is that, a PC will be placed at Base workshops which shall handle all in coming (Units who wishes to send or check the status of repair), Telephone calls and reply to the units according to their queries, after searching the data base. The PC will search for the requested data, convert the information in voice form and sent it to the requesting unit over the telephone line. It is appreciated that by establishing such a system units could remotely access and retrieve the required information about their vehicles / equipment, through a telephone line. However this is a research oriented project basing on the research a prototype of the project will be developed. All necessary implementation parameters could be included for actual implementation.

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 Background:

Corps of Electrical and Mechanical Engineers (E.M.E) is one of the administrative services which plays its part in administration of the Army by providing a comprehensive repair and maintenance supports to vehicles and equipment used by the units of Pakistan Army. The task of repair and maintenance is performed at various echelons of repair. At field formation level this task is being performed by field workshops. Field work shops falls under second echelon of repair and according to the permissive repair schedule issued by General Head Quarters (G.H.Q) field workshops are authorize to carry out limited repair work.. The work that does not fall with in the permissive repair schedule of field workshops is forwarded to higher echelons of repair on deposit repair work orders. This includes Area, Medium and Base workshops. E.M.E Base Workshops are designed to carry out base repair and overhaul of repairable stock held by the ordnance depots. Base workshops play an important role in managing requisite level of vehicles / equipment availability to the field units of Pakistan Army. 501 Central workshop E.M.E is the largest Base workshop of Pakistan Army. Basic aim of this workshop is to deal in overhauling of Vehicles. G.H.Q assignees various overhauling projects to this workshop in order thin out the repairable stock of Central Mechanical Transport and Supply Depots (C.M.T&S.D) Golra. As a result of these projects bulk volume of repairable stock is feed to 501 central workshop. On the other hand field formations also sends their vehicles after initiating and approval of deposit repair work orders to get them overhauled. So there are two types of clients that deal with Base

workshops i.e. G.H.Q and field units. Both these clients need to establish a close communication link with base workshop to monitor the progress of repair work being done at base workshop. Field units located far from base workshop, establishes this link through traditional mailing system or by sending liaison parties which is a time consuming process.

1.2 Existing procedure:

Action for deposit repair will normally be initiated, when a vehicle, has gone out of action, is beyond the capability of field workshop. User units for sending the vehicle to the base workshops are adopting the following procedures.

a. Units wish to send their vehicle / equipment for base repair, they initiates the deposit repair work order for base workshop through CEME and GHQ, EME Directorate.

b. On receipt of deposit repair work order, G.H.Q asks the feasibility of repair from base workshop concern, to accord sanction of repair.

c. Base workshop after carrying out necessary ground checks keeping in view the factors such as availability of floor space in workshop, existing workshop commitments communicate a PSS (please send store) date to the units.

d. Once the deposit repair work order get sanctioned and Units receives PSS date they send their vehicles to base workshops.

e. After depositing the vehicle to the base workshops, units start keeping track to monitor the progress and completion of the job. This leads to a lengthy correspondence between units and base workshops. In case some user unit wants to check the status of their

vehicle on telephone Planning and Control (P & C) section, due to heavy repair commitments in overhauling of depot stock, do not have the current repair situation of all field repair works., so the units are asked to call again after some delay.

f. For monitoring the status of overhauling projects G.H.Q and C.M.T&S.D Golra are dependent on monthly report and returns. If in case they want to know latest position they have to contact to the officer in charge P&C, who after consulting his record will reply.

1.3 Problem Statement:

As mentioned those base workshops are aimed to provide overhauling facilities to the vehicles and equipment. The base workshop chosen as reference workshop for the project is the largest base workshop vis-a-vis it's dependent units are in huge number. Situation arises when all the units start asking about their vehicle's repair status and at the same time some other units start requesting for a vacancy in order to deposit their vehicle. The officer in charge P&C-section of base workshop handles reply to all those queries. The existing system has a problem that before depositing the vehicles field units has to wait a considerable long time to know the availability of floor space in base workshop. Second after depositing the vehicle they either check the status of repair through traditional mail system or by engaging officer in charge P&C on telephone and depriving him from his basic planning task by calling number of times. Existing system not only time cumbersome for the base workshops but also irritating for the user units.

1.4 Problem solving approach:

The concept of the project is discussed with the help of following flow chart. The flow diagram gives a bird's eye view about the basic building blocks used in the project and the sequence of execution. Once the user will call from a remote location he gets connected with a PC. Before login the caller PC will first ask the user for his password. Only valid users will be welcomed to proceed with the query. This activity of project will be dealt through Telephony Application Programming Interface (TAPI), by utilizing Modem and sound card. Database is designed using Microsoft Access. The complete code of the project is written in Microsoft Visual C++.

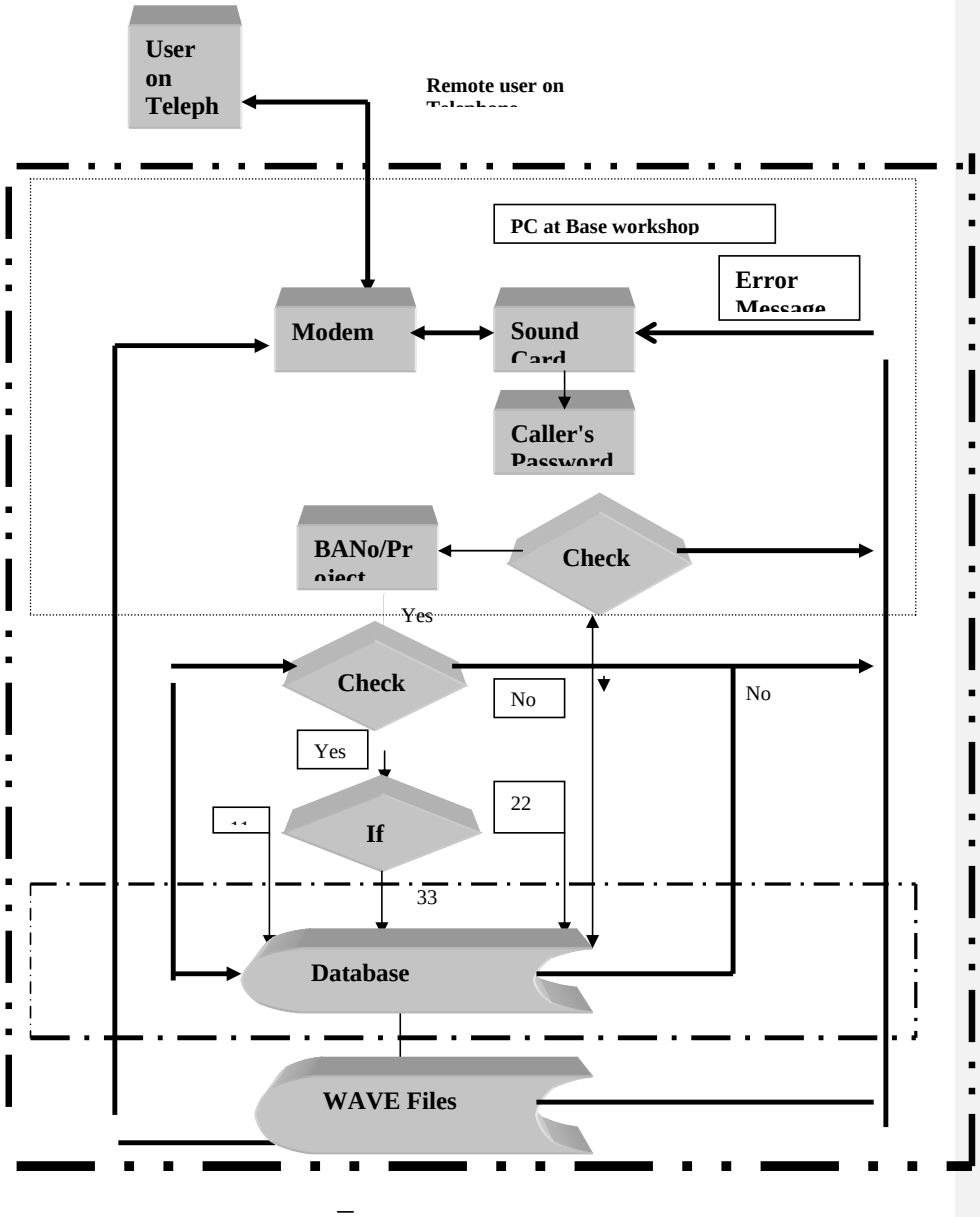

Fig 1.1 Concept of Project

1.5 Objective and advantages:

Officer in charge of P&C is the key person in any base workshop. He has to plan the working of base workshop in such a fashion that all available resources, such as workshop floor capacity, Employment of technicians and availability of required spare parts are utilized optimally, and no work get pending for want of theses resources. It is also the part of the planning that workshop should remain functional through out the year. Keeping in view the quantum of job, performed by the officer in charge P&C, it is felt that some system must exists that not only provide timely assistance and current situation to the user units but on the other hand give some relief to the officer in charge P&C to plan the things smoothly. Under this scenario the objective of this project is, *"To design a prototype of automated data retrieval system at base workshop. That may provide current repair status of the vehicles belonging to field units and C.M.T&S.D Golra vis-a-vis communicate the availability or non availability of floor space before accepting a new vehicle for overhauling*". By designing such a automated system G.H.Q, C.M.T&S.D Golra and field units could get the required information from the computer placed at base workshop through telephone with out engaging a person at the base workshop's end. Interested person will dial the number of base workshop, that telephone line will be connected to a PC which offer the caller to login through entering his pass word and then invites to enter the desired query. The PC will reply to the caller an appropriate message based on the query entered by the caller. During the entire process PC at base workshop will be handling the telephonic calls by searching the database. That caller need not to be in possession of PC, instead he will be getting the required information on his telephone set.

Keeping in view the existing procedure, following advantages will lead and to enhance the efficiency of P&C as well in saving the time and efforts of user units.

a. Field units will get timely information about availability of repair vacancy. If the vacancy exists, the PC will communicate a control number against that particular vacancy. Basing on that control number the units may start completing the required formalities of depositing their vehicle this will cut down the delay in waiting for reply.

b. Field units will get the current information about the status of their deposited vehicles at any time they wish to know about. The caller himself will retrieve that information from the PC through telephone. This give relief to the user units as well as to the person concern at base workshop.

c. It would ease out the working at Base workshops by providing relief to the officer in charge of P&C-section by not addressing the correspondence regarding field units queries.

d. It would facilitates the GHQ (Ordinance Directorate, EME Directorate) and ordinance depots to coordinate with Base workshops more efficiently, as they will begetting the current information at any time during the phase of their planning.

1.6 Organization of Dissertation

(This portion will be completed after finalization of written material)

CHAPTER 2

Telephony API

2.1 Introduction

When the term telephony is heard in the context of computer, it gives a thought ~~he think~~ of data or FAX modems and voice grade telephone lines. ~~and very little else.~~ Telephony Application Programming Interface (TAPI) goes far beyond these simple concepts and provides a consistent programming interface for a variety of devices operating on voice grade lines. The devices include modems, FAX modems, voice capable modems, computer-controlled telephone sets, and many more. TAPI provides services for placing outgoing calls, accepting incoming calls, and managing calls and devices. This chapter provides a general overview of the T~~elephony~~ API and how it fits into the Windows Open Services Architecture (WOSA) ~~(Windows Open Services Architecture)~~ model. Two main devices, Line devices and Phone devices, defined within the TAPI model will be discussed in detail.

2.2 Definitions

2.2.1 Telephony API

The Telephony Application Programming Interface (TAPI) is an application-programming interface that is used to communicate by means of telephones. The TAPI is one of the most significant API sets released by Microsoft, API is a set of routines that an application program uses to request and carry out lower-level services performed by a computer's operating system. It is a single set of function calls that allows programmers to manage and manipulate any type of communications link between the PC and the telephone line [1].

Formatted

Formatted

Formatted

Formatted

Telephony services are divided into Assisted Telephony services and the services provided by the full Telephony API. In general, the full Telephony API is used to implement powerful telephonic applications and Assisted Telephony is used to add minimal but useful telephonic functionality to non-telephony applications. Telephony's services are divided into the groups shown in the following illustration:

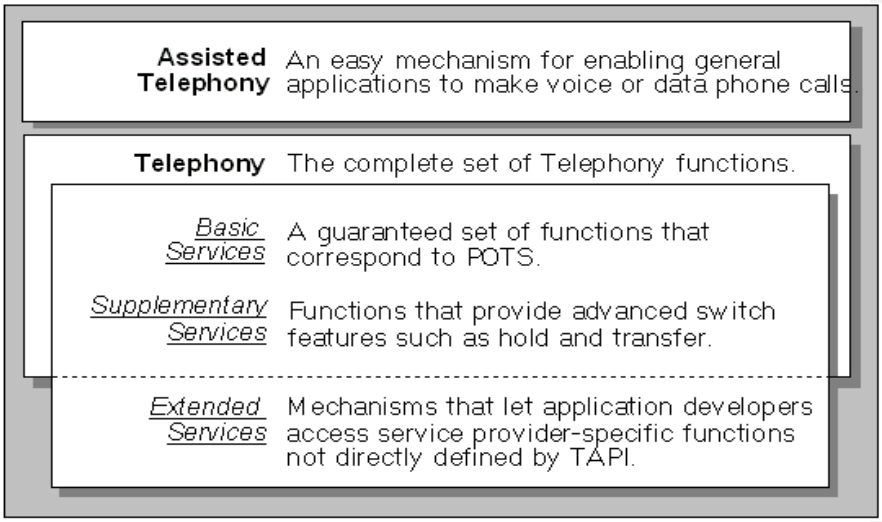

Fig 2.2 Grouping of TAPI services

2.2.2 Windows Open System Architecture(WOSA)

Formatted

Based on the WOSA model, Windows Telephony consists of the TAPI and TAPI32 dll, (which forward application requests to the telephony service for processing), tapisrv.exe (which implements and manages the TAPI functions), and one or more telephony service providers (drivers).

When an application calls a TAPI function, the TAPI dynamic-link library validates and marshals the parameters of the function and forwards it to tapisrv.exe details about DLL and it’s advantages in use

is shown at Appendices ‘A’. TAPISRV (the telephony service) processes the call and routes a request to the appropriate service provider. To receive requests from TAPISRV, the service provider must implement the Telephony Service Provider Interface (TSPI). A user can install any number of service providers on a computer as long as the service providers do not attempt to access the same hardware device at the same time. The user associates the hardware and the service provider when installing. Some service providers may be capable of accessing multiple devices. In some cases, the user may need to install a device driver along with the service provider. Applications use the TAPI functions to determine which services are available on the given computer. TAPI determines what service providers are available and provides information about their capabilities to the applications. In this way, any number of applications can request services from the same service provider; TAPI manages all access to the service provider. As long as an application does not depend on optional features, the applications can, without modification, use any services to carry out telephony tasks, even services made available after the application is developed. This is because the application always accesses the many different services through TAPI, which translates the requests the application makes into the actual protocols and interfaces required [2].

2.2.3 Modem

Formatted

At a minimum, the modem must support the Microsoft Windows 98 operating system, including TAPI and communications under the Microsoft Win32 API. Most modems are compatible with Windows 98.A modem is a device used for data or fax transmission. If Windows 98 does not support a modem, or the modem is not directly compatible with a supported model, the modem manufacturer must supply a driver

for the modem. This driver must supply the Service Provider Interface (SPI) functions called by TAPI. Most commercially available modems comply with Windows 98, because almost all comply closely with the international standards to use the UNIMODEM service provider included in Windows 98 [3].

2.2.3.1 Unimodem

The universal modem driver, provided with Windows, that translates Telephony Service Provider Interface (TSPI) calls into AT commands, and sends the commands to a virtual device driver that talks to the modem. . A universal modem is the one that supports standard modem AT commands, Such as PCMCIA modems.

a) Wave Files

2.3 The Telephony API Model

The telephony API model is designed to provide an abstracted layer for access to telephone services on all Windows platforms. In other words, the telephony API is a single set of functions that can be used to access all aspects of telephony services within the Windows operating system. The aim of TAPI is to allow programmers to write applications. Applications written using TAPI to gain direct access to telephone-line services work the same on analog or digital phone lines. Applications that use TAPI can generate a full set of dialing tones and flash-hook functions. The TAPI design model is divided into two areas, each with its own set of API calls [4]. The two TAPI devices are, *Line devices,* to model the physical telephony lines used to send and receive voice and data between locations and *Phone devices* to model the desktop handset used to place and receive calls.

2.3.1 Line Devices

Formatted

The line device is used to model the physical telephone line. It is important to understand that, in TAPI, the line device is not really a physical line; it's just a model or object representing a physical line. In TAPI applications, a program could keep track of several line devices, each of which is connected to a physical line. That same TAPI application could also keep track of multiple line devices that number more than the total physical lines available to the PC. For example, a single TAPI application could be designed to provide voice, fax, and data links for a user. The TAPI application would identify three line devices. One for voice calls, one for fax transmission, and one for sending and receiving data via an attached modem. If the PC has only one physical phone line attached, the TAPI application would share the one line between the three defined line devices. This is called dynamic line mapping. Each time the TAPI application starts a line device, it requests the first available physical line that has the capabilities needed (voice, fax, data, and so on). If a line is not available, a message to that effect is returned to the calling program. In some cases, such as fax transmissions, the TAPI application may "queue up" the line request for processing at a later time. If two lines are available, the TAPI application uses them as they are needed. If a third line device becomes active, the TAPI application knows that there are no other available open lines and notifies the user (or possibly queues up the outbound call for later) [5].

2.3.2 Phone Devices

The second type of device modeled by TAPI is the phone device. This model allows TAPI programmers to easily create "virtual phones" within the PC workspace. For example, a standard PC with a sound card, speakers, and microphone can emulate all the functions of a desktop phone. These virtual phones, like their line device

counterparts, need not exist in a one-to-one relationship to physical phones. A single PC could model several phone devices, each with their own unique characteristics. When an actual call must be made, the user could select one of the phone devices, enter the desired number and then the TAPI application would attach the phone device to an available line device [6].

2.4 Typical Configurations

The TAPI model is designed to function in several different physical configurations, which each have advantages and drawbacks. There are four general physical configurations [7]:

2.4.1 Phone-based

This configuration is best for voice-oriented call processing where the standard handset (or some variation) is used most frequently. In phone-based TAPI configurations, the standard telephone handset is connected to the telephone switch and the PC is connected to the telephone. This configuration is most useful when the telephone handset is the primary device for accessing the telephone line. Since the telephone rests between the PC and the switch, the PC may not be able to share in all the activity on the line. A phone-based configuration does not preclude the use of the PC to originate calls. As long as the PC is equipped with a phone card that allows dialing, the PC can originate a call and then allow the handset to pick up on that call at any time

2.4.2 PC-based

This configuration is best for data-oriented call processing where the PC is used most frequently for either voice or data processing. Shared or unified line-This is a compromise between phone-based and PC-based systems. It allows all devices to operate as equals along the service line. The primary difference between this configuration and the

others is that the PC acts as either a voice-server or a call switching center that connects the outside phone lines to one or more PCs and telephone handsets. The primary advantage of multiline configurations is that user do not need a direct one-to-one relationship between phone lines and end devices (phones or PCs). PC-based TAPI configurations place the PC between the telephone switch and the standard handset. This configuration is most useful when the PC is the primary device for accessing the telephone line. In this configuration, the PC most often originates phone calls. Typically, this is done via a phone card and software on the PC that manages a list of phone numbers and handles the dialing of the phone. Depending on the exact media mode of the call, the PC can be used to display digital data on screen while handling voice information, too. Users could originate a voice call through the handset and then switch to the PC to capture and display digital data sent over the same line. Another major advantage of the PC-based configuration is that the PC can act as a call manager for the handset. This is especially valuable in a mixed-mode environment where voice, data, and fax are all coming in to the same phone address. For example, as a call comes in to the

attached phone line, the PC can answer the call and determine the media mode of the call. If it is a fax call, the PC can route the call directly to an attached fax machine (or to the fax driver on the PC). Data calls can be handled directly by the PC and voice calls can be forwarded to the attached handset.

2.4.3 The Shared or Unified Line Configuration

The shared or unified line configuration is a bit of a compromise between PC-based and phone-based configurations. The shared line configuration involves a split along the line leading to the switch. Both the PC and the phone have equal (and simultaneous) access to the line.

The advantage of the shared-line configuration is that either device can act as the originator of a call. The primary disadvantage is that both devices have equal access to incoming calls. In other words, as a call comes in, both devices will ring. Depending on the software operating on the PC, it is possible that both devices would attempt to answer the same incoming call. This situation is much like having two extension phones along the same access line. The unified line configuration offers the combined benefits of the PC-based configuration and the shared-line configuration. In the unified line configuration, the access line goes directly from the switch into a telephone card in the PC. The PC also has handset equipment either attached to the phone card or integrated into the PC itself.

2.4.4 Multilane Configuration

TAPI is also designed to support multiline configurations. In this arrangement, TAPI is used to provide third-party call control. Single lines act as the first (and only) party in a telephone call. In a multiline environment, a device can act as a third party in a telephone call. The most common form of third-party call control is a central switchboard in an office. When a call comes in, the switchboard (the third party) accepts the call, determines the final destination of the call, routes the call to the correct extension, and then drops out of the call stream.

2.5 Architectural Overview

Win32 Telephony is a full, 32-bit implementation of the original 16-bit Windows Telephony application and service provider interface. Other than components provided for backward compatibility, all components of Win32 Telephony, including service providers, are implemented in 32 bits. The following figure illustrates the architecture of Win32 Telephony on the Windows operating systems [8].

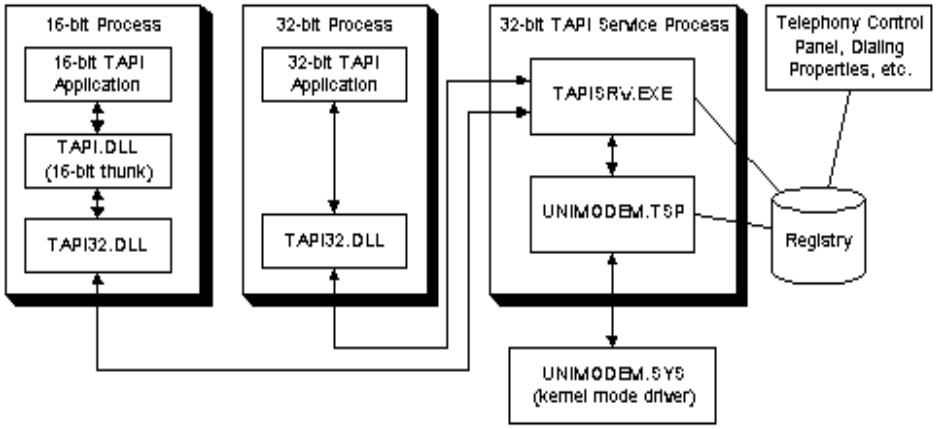

Fig 2.2 TAPI Architecture

Existing 16-bit applications link to TAPI. In Windows 3.1 and Windows 95, TAPI is the core of Windows Telephony. Under Windows NT/Windows 2000, TAPI is simply a thunk layer to map 16-bit addresses to 32-bit addresses, and pass requests along to tapi32.dll. (Thunk is a small section of code that performs a translation or conversion during a call or indirection. For example, a thunk is used to change the size or type of function parameters when calling between 16-bit and 32-bit code.)

Existing 32-bit applications link to tapi32.dll. In Windows 95, tapi32.dll is a thunk layer to TAPI. In Windows 98 and Windows NT/Windows 2000, tapi32.dll is a thin marshaling layer that transfers function requests to tapisrv.exe and, when needed, loads and invokes service provider user interface DLLs in the application's process,

(Marshaling is defines as the packaging and sending interface method calls across thread or process boundaries).

In the Win32 implementation, tapisrv.exe is the *core* of TAPI. It runs as a separate *service process*, subsuming the role played in Windows 3.1 and Windows 95 by the tapisrv.exe hidden process and TAPI itself. All telephony service providers execute in the tapisrv.exe process, eliminating the difficulties that arose in previous versions of Windows telephony caused by service providers sometimes running in the TAPIEXE context, and sometimes in the context of an individual application. Service providers can create threads in the TAPISRV context as needed to do their work, and be confident that none of the resources they create will be destroyed by the exit of any individual application.

Underneath the telephony service provider DLL, the service provider can use any system functions or other components necessary. These functions include CreateFile and DeviceIOControl, which work with independent hardware vendor-designed kernel mode components and services, as well as standard devices such as serial and parallel ports to control external, locally attached devices. The Telephony service provider user interface DLL is new. This DLL is loaded by TAPI into the process of an application that invokes any of the service provider functions that can display a dialog box (for example, TSPI_lineConfigDialog). The service provider can also cause its associated UI DLL to be loaded and executed in the process of an application if the service provider needs to display UI at unexpected times, such as to display the Talk/Hang-up dialog box displayed by the Universal Modem Driver (UNIMODEM) when a data modem is used to dial an interactive voice call using TSPI_lineMakeCall (not normally considered to be a UI-generating function).

2.6 TAPI Software Architecture

The heart of TAPI is the TAPI dynamic link library (DLL) that offers TAPI services to the applications. This DLL serves as a layer between telephony applications and TAPI service providers. One such service provider is the UNIMODEM driver; this Universal Modem driver is supplied with Windows 95 and provides TAPI services for modems compatible with the Hayes AT command set [6]. This basic TAPI architecture is shown in Fig 2.2. In addition to the TAPI DLL and the telephony service providers (drivers), another important, albeit invisible, component of TAPI is the executable program tapiexe.exe. This program plays an important role when TAPI sends notifications to the calling application via callback functions [9].

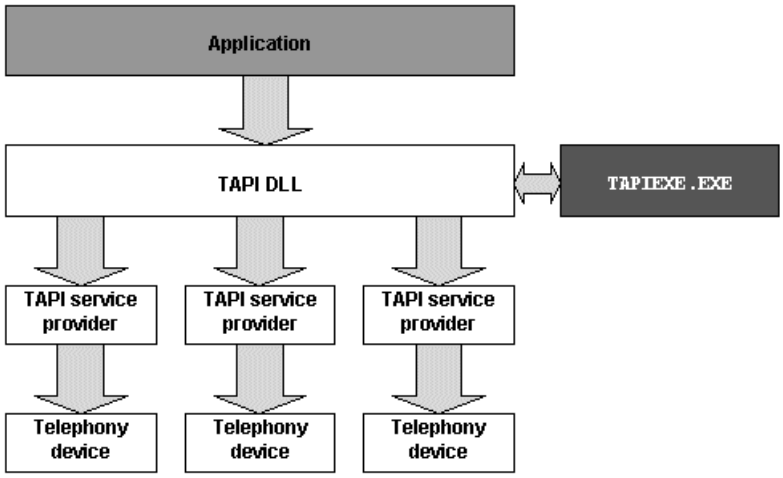

Fig 2.2 The TAPI software architecture.

2.7 Synchronous and Asynchronous Operations

Many TAPI operations are synchronous; that is, when the TAPI function returns, the operation is either completed or failed, in which case an error code is returned. However, some TAPI operations are asynchronous; the TAPI function returns indicating whether the TAPI operation has been successfully initiated, but the operation is

completed in another thread, and the application is notified via a callback function. The callback function is registered with TAPI when the TAPI library is initialized [10]. The actual callback mechanism deserves a closer examination, especially because it has some consequences as to how TAPI functions operate. When a service provider wishes to place a notification, it calls the TAPI DLL. In effect, it requests that the DLL notify all concerned applications that a specific event has taken place. This first call to the TAPI DLL takes place in the execution context of the service provider. The TAPI DLL in turn sends a message to tapiexe.exe. This executable program calls the TAPI DLL itself, this time in its own execution context. This call instructs the TAPI DLL to post a Windows message to the applications that need to be modified. When the application receives and processes the message in its message loop, the message is dispatched to the TAPI DLL again, this time in the application's execution context. The TAPI DLL may in turn call the application's registered TAPI callback function to notify the application of a TAPI event. The TAPI notification mechanism is shown in Fig 2.4

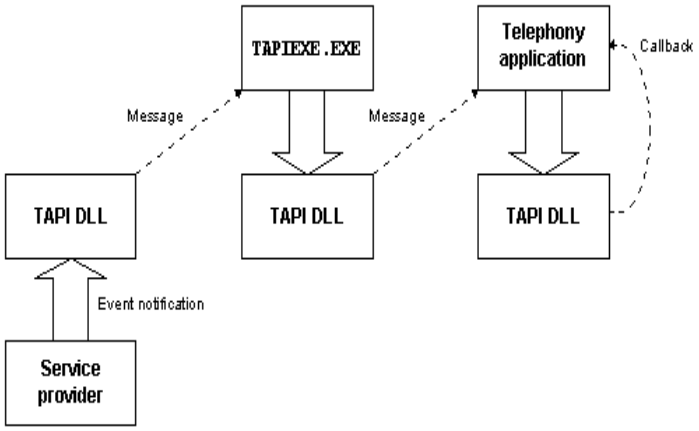

Fig2.4 Processing of TAPI events.

This mechanism has important implications for the architecture of TAPI applications. For one thing, the scenario described here makes it clear that TAPI applications must have a message loop in order to process notifications correctly. Although the use of a callback function may imply that a message loop is unnecessary, this is not the case; the callback function is only called after the application receives a Windows message that the TAPI DLL processes. Another consequence concerns the use of multiple threads. It is important to realize that in order for TAPI to operate as expected threads that call asynchronous TAPI functions must have a message loop. The callback function is called in the context of the thread making the asynchronous call; this cannot happen unless the thread processes Windows messages.

2.8 Summary

TAPI is comparatively new subject and it has an important role in implementation of this project. Secondly this subject has not yet touched during any of the previous courses. Keeping in view its

applications and it support in developing telephony based projects, it is discussed in detail, so that it give benefit in future research.

2.9 Chapter's Bibliography

1. Telephony API
2. WOSA
3. Modem
4. TAPI
5. Line devices
6. Phone device
7. Typical configuration
8. Architectural over view
9. Software architecture
10. Synchronous and Asynchronous Ops

CHAPTER 3

Design Parameters

3.1 Introduction

Before designing the database the understanding of design and manipulation technique is an important factor which must be catered for. The factors considered with Visual C++ are briefly discussed here. Visual C++ is a vast programming language so only the portion related to databases are discussed in this chapter. It focuses on the definitions, techniques and methodologies used to design a database. The tables for the project are designed after considering these design parameters.

3.2 Database

3.2.1 Table

A database consists of one or more tables; if more than one table is included in a database, the entities described in the tables must be related by at least one attribute class (field) that is common to two of the tables. Tables for the project are designed basing on this basic rule. Detailed discussion about the database of the project will be found in chapter 4. Database of the project consists of more then one table as shown in the following figure.

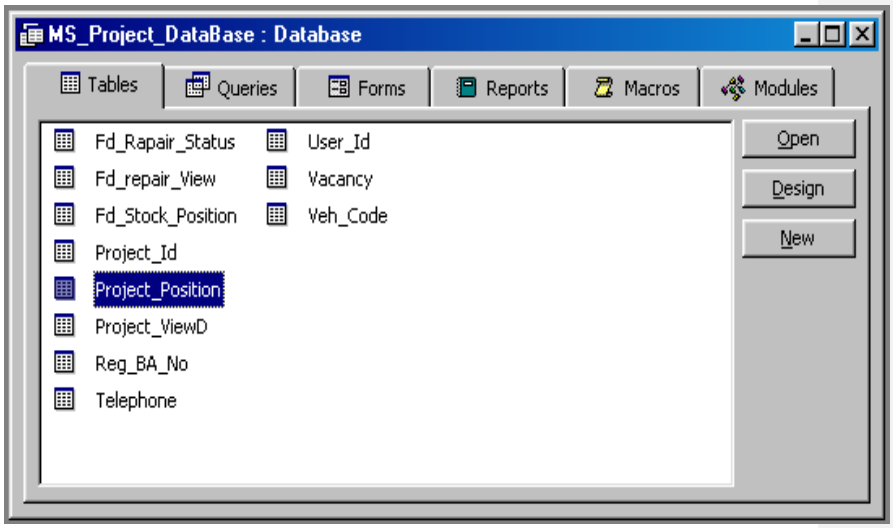

Fig 3.1 Project database tables

3.2.2 Query

A query is a method by which programmers can obtain access to a subset of records from one or more tables that have attribute values satisfying one or more criteria. There are a variety of ways to process queries against database [11]

3.2.3 Relations

The relational database models define three types of relations:

a. **One-to-one relations** require that one and only one record in a dependent table relate to a record in a primary table. One-to-one relations are relatively uncommon in relational databases.

b. **One-to-many relations** let more than one dependent table relate to a record in a primary table. The term many-to-one is also used to describe one-to-many relations. One-to-many relations constitute the relational database model's answer to the repeating-groups problem. Repeating groups are converted to individual records in the table on the "many" side of the relation. One-to-many relations are the most common kind of relations.

c. **Many-to-many relations** aren't true relations, because many-to-many relations between two tables require an intervening table, called a relation table, to hold the values of the foreign keys. (Relational-database theory only defines relations between two tables.)

Keeping in view these parameters the queries are created for the project. Following figure shows the design view of one of the query designed to handle a query based on three different tables.

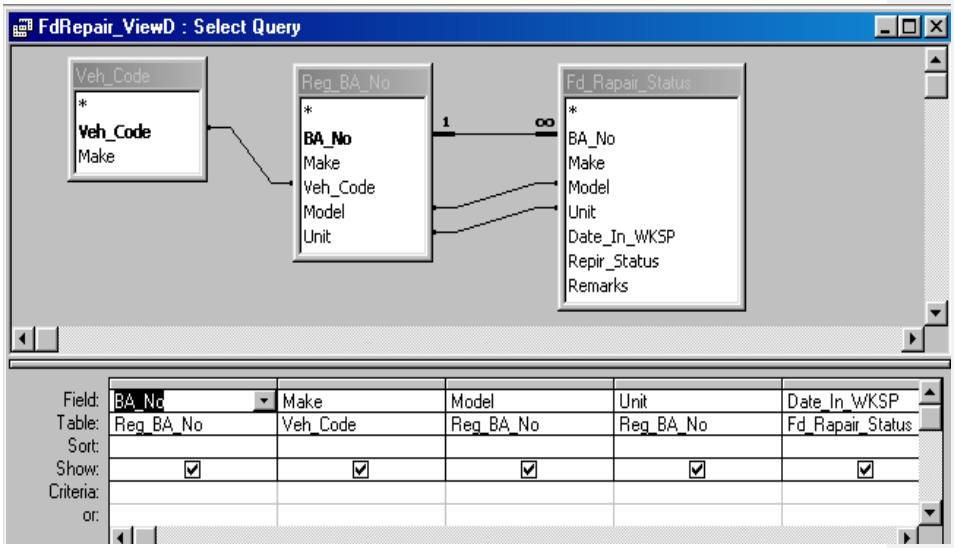

Fig 3.2 Query Design

3.2.4 Normalizing Table Data

The process of transforming existing data into relational form is called normalization. Normalization of data is based on the assumption that the data has been organized into a tabular structure wherein the tables contain only a single entity class [12]. The objectives of normalization of data include the following:

a. Eliminating duplicated information contained in tables

b. Accommodating future changes to the structure of tables

c. Minimizing the impact of changes to database structure on the front-end applications that process the data

The following sections describe the five steps that constitute full normalization of relational tables. In most cases, the process of normalization is halt at third normal form, or model. Many developers bypass fourth and fifth normal forms because these normalization rules appear arcane and inapplicable to everyday database design.

3.2.4.1 First Normal Form

First normal form requires that tables be flat and contain no repeating groups. A data cell of a flat table may contain only one atomic (indivisible) data value. If imported data contains multiple data items in a single field, then one or more new fields need to be added to contain each data item and then move the multiple data items into the new field.

3.2.4.2 Second Normal Form

Second normal form requires that all data in nonkey fields of a table be fully dependent on the primary key and on each element (field) of the primary key when the primary key is a composite primary key.

3.4.2.3 Third Normal Form

Third normal form requires that all nonkey fields of a table be dependent on the table's primary key and independent of one another. Thus, the data in a table must be normalized to second normal form in order to assure dependency on the primary key. The issue here is the dependencies of nonkey fields. A field is dependent on another field if a change in the value of one nonkey field forces a change in the value of another nonkey field.

3.4.2.4 Over-Normalizing Data and Performance Considerations

After it has been determined that data structure meets third normal form, the most important consideration is to avoid over-normalizing the data. Over-normalization is the result of applying too strict an interpretation of dependency at the third normal stage. Creating separate tables is then the solution and these tables need to be joining these tables in a one-to-one relationship to display the data values This will be a very slow process unless indexes are created on the primary key of each table.

3.4.2.5 Fourth Normal Form

Fourth normal form requires that independent data entities not be stored in the same table when many-to-many relations exist between these entities. If many-to-many relations exist between data entities, the entities aren't truly independent, so such tables usually fail the third normal form test. Fourth normal form requires to create a relation table that contains any data entities that have many-to-many relations with other tables.

3.4.2.6 Fifth Normal Form

Fifth normal form requires that to exactly reconstruct the original table from the new table into which the original table was decomposed or transformed. Applying fifth normal form to resulting table is a good test to make sure that data didn't lose in the process of decomposition or transformation.

Under the guidance of these design parameters a stable set of relations is created to handle and manipulate the required data. Keeping in view all these parameters and by applying all normalization techniques database tables are created with reduced redundancies and inconsistencies. Following figure shows one of the tables, which is created after considering all the design parameters.

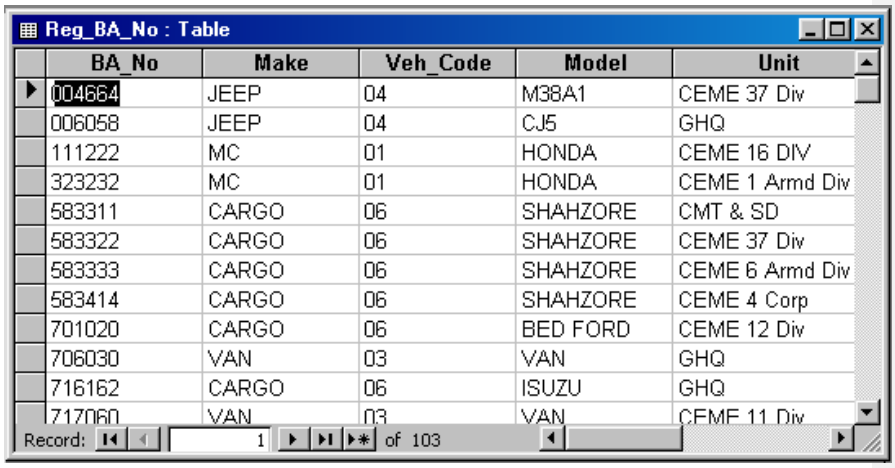

Fig 3.3 Normalized database table

3.2.5 Jet Database Engine

The Microsoft Jet Database Engine is defined as a database management system that retrieves data from and stores data in user and system databases. The Microsoft Jet database engine can be thought of as a data manager component with which other data access systems, such as Microsoft Access and Microsoft Visual Basic, are built. Microsoft Jet provides a variety of database management services, including data definition, data manipulation, data integrity, and security [13]. There are six basic services that a DBMS should provide. The six basic functions of a DBMS are discussed as under:

a. **Data definition and integrity**: create and modify structures for holding data, such as tables and fields, and ensure that rules to prevent data Corruption from invalid entries or operations are applied to data operations.

b. **. Data Storage:** Stores data in the file system.

c. **Data manipulation:** Add new data, modify or delete existing data.

d. **Data retrieval:** Retrieve data from the system**.**

e. Security: Control users' access to database objects and data, there by protecting the objects and data against unauthorized use.

f. Data sharing: share data in a multi-user environment.

3.2.6 Data Accesses Object(DAO) Vs Open Database Connectivity(ODBC)

DAO enable the programmers to access and manipulate databases through the Microsoft Jet database engine. Through this engine, programmers can access data in Microsoft Access database files. Visual C++ provides extensive support for building DAO applications through its feature AppWizard. Both DAO and ODBC are application programming interfaces (APIs) that provides the ability to write applications that are independent of any particular database management system (DBMS) [14]. The choice between ODBC and DAO is often a difficult one. In general, DAO provides more flexibility, with support for both Data Definition Language (DDL) and Data Manipulation Language (DML). The differences between DAO and ODBC .are highlighted in the following table

Table: 3.1 ODBC Vs DAO

	DAO Classes	**ODBC Classes**
Access MDB files	Yes	Yes
Data Sources	Yes	Yes
Available for 16 bit	No	Yes
Available for 32 bit	Yes	Yes
Database Compaction	Yes	No
Database Engine Support	Microsoft Jet Engine	Target Database
DDL Support	Yes	Only via ODBC Call
DML Support	Yes	Yes
Updateable Joins	Yes	No

3.3 Microsoft Foundation Classes (MFC)

The MFC Library is the most distinguishing component of the Visual C++ development system. This vast collection of C++ classes and the hierarchical representation is shown as per appendix ‘C’. MFC encapsulates most of the Win32 API and provides a powerful framework for typical applications. A typical MFC application is one that is created by using the Visual C++ AppWizard [15]. The primary goal of the MFC is to provide an encapsulation for the Windows API. Ideally, an MFC application never has to call Windows API functions directly; instead, it constructs an object of the appropriate type and utilizes the object's member functions. take care of any initialization and cleanup that is necessary. For example, an application that needs to draw into a window can do so by constructing a CClientDC object and calling the object's member functions. The CClientDC constructor makes the appropriate calls to create a device context, set up the mapping mode, and perform other initializations. When the object goes out of scope or is destroyed using the delete operator, the destructor

automatically releases the device context. This kind of encapsulation would make writing application programs easier. The classes in MFC are loosely organized into several major categories as depicted in appendix c.

3.4 The Data Access Objects of Visual C++

Database objects have existed in MFC and Visual C++. Microsoft Jet database engine, combined with database functions incorporated in the Visual C++. DAO lets the developers to create database objects using tables native to any of the more common desktop and client-server RDBMSs. In addition, Visual C++ lets the developer define and create new databases for the majority of the supported database types, [16]. Following are the objects that are contained in MFC and Visual C++:

a. **CDatabase** objects function as the linkage between the application and the actual dataset. In C programs, the functionality of the CDatabase object is available using the SQL...() functions.
b. **CRecordset** objects represent the results, or set of records, obtained from a dataset. The CRecordset object contains CDatabase tables contained in the CDatabase object.
c. **CRecordView** objects are based on the CFormView class. With CFormView, the application functions much like any other dialog-based application.

While dealing the database with Visual C++ few terms are used very frequently these terms are described in the following paragraph.

3.4.1 Rrecordset

A recordset, is an object that contains a set of records from the database. Some recordset types are based on single tables. More commonly, however, recordsets are based on a query that is created by

using Structured Query Language (SQL). Whereas a table-type recordset is based on a single table, an SQL-based query can be used to retrieve a specific subset of records, or to retrieve information from two or more tables at once. SQL enables the user to specify which records to retrieve and the order in which to retrieve them [17].

3.4.2 Dynaset

Dynaset-type record sets provide more flexibility than tables. A Dynaset can refer to any table, attached table, or query. A Dynaset provides an updateable view to the data. As the application scrolls to a changed record, a new copy is retrieved, bringing it up to date. This dynamic behavior is ideal for situations in which it is important to be completely up to date. A Dynaset contains a set of keys to the underlying tables [18]. The actual data is retrieved when the user accesses a particular record. Because the user use a key set to refer to the data, Dynaset reflects modifications to existing records by other users, but does not reflect new records added by other users.

3.4.3 Snapshot.

A "snapshot" reflects the state of the data at a particular moment, the moment the snapshot is taken. This behavior is ideal for reporting. Because it takes time to retrieve the records for a snapshot, the moment at which the snapshot occurs is not instantaneous [19].

3.4.4 Table.

A table-type recordset is based directly upon the table rather than on a query. The table-type recordset corresponds to a single table and can be updated. The records in a table-type recordset always reflect all changes that are made to that table, including the additions or deletions made by other users [20].

3.5 Summary

This chapter introduced the methodology of designing efficient relational database structures, including modeling tools for Jet databases and normalizing tables that are created from existing data sources. Entire chapter is devoted to database design techniques, and the MFC classes used in Visual C++ to access, manipulate and retrieve data to and from database tables.

Chapter's Bibliography

11. Queries

12. Normalization

13. Jet database engine

14. DAO Vs ODOB

15. MFC Classes

16. DAO of Visual C++

17. Recordset

18. Dynaset

19. Snapshot

20. Table

Chapter 3, Appendix 'C'

MFC Library and Hierarchy Chart

C.1 MFC Library:

The library's classes are presented here in the following categories:

(1) .**Root Class:** CObject

(2) **MFC Application Architecture Classes**

- a. Application and Thread Support Classes
- b Command Routing Classes
- c. Document Classes
- d. View Classes (Architecture)
- e. Frame Window Classes (Architecture)
- f. Document-Template Classes

(3) **Window, Dialog, and Control Classes**

- a. Frame Window Classes (Windows)
- b. View Classes (Windows)
- c. Dialog Box Classes
- d. Control Classes
- e. Control Bar Classes

(4) **Drawing and Printing Classes**

- a. Output (Device Context) Classes

- b. Drawing Tool Classes

(5) **Simple Data Type Classes**

(6) **Array, List, and Map Classes**

(7) **Template Classes for Arrays, Lists, and Maps**

a. Ready-to-Use Array Classes

b. Ready-to-Use List Classes

c. Ready-to-Use Map Classes

d. File and Database Classes

e. File I/O Classes

(8) **DAO Classes**

(9) **ODBC Classes**

(10) **Internet and Networking Classes**

(11) **OLE-Related Classes**

(12) **Debugging and Exception Classes**

(13) **Debugging Support Classes**

(14) **Exception Classes**

C.2 MFC Hierarchy:

The hierarchical representation of MFC classes is reflected on next page.

B.1 What Are DAO and ODBC?

Both Data Access Objects (DAO) and Open Database Connectivity (ODBC) are application programming interfaces (APIs) that give you the ability to write applications that are independent of any particular database management system (DBMS).

DAO is familiar to database programmers using Microsoft Access Basic or Microsoft Visual Basic. DAO uses the Microsoft Jet database engine to provide a set of data access objects: database objects, tabledef and querydef objects, recordset objects, and others. DAO works best with .MDB files like those created by Microsoft Access, but you can also access ODBC data sources through DAO and the Microsoft Jet database engine.

ODBC provides an API that different database vendors implement via ODBC drivers specific to a particular database management system (DBMS). Your program uses this API to call the ODBC Driver Manager, which passes the calls to the appropriate driver. The driver, in turn, interacts with the DBMS using Structured Query Language (SQL).

B.2 How MFC Encapsulates DAO

The MFC DAO classes treat DAO much as the MFC classes for programming Windows treat the Windows API: MFC encapsulates, or "wraps," DAO functionality in a number of classes that correspond closely to DAO objects. Class CDaoWorkspace encapsulates the DAO workspace object, class CDaoRecordset encapsulates the DAO recordset object, class CDaoDatabase_encapsulates the DAO database object, and so on.

MFC's encapsulation of DAO is thorough, but it is not completely one-for-one. Most major DAO objects do correspond to an

MFC class, and the classes supply generally thorough access to the underlying DAO object's properties and methods. But some DAO objects, including fields, indexes, parameters, and relations, do not. Instead, the appropriate MFC class provides an interface, via member functions, through which they can be accessed for example:

a. The fields of a recordset object
b. The indexes or fields of a table
c. The parameters of a querydef
d. The relations defined between tables in a database

A.1 DLLs: Overview

A dynamic-link library (DLL) is an executable file that acts as a shared library of functions. Dynamic linking provides a way for a process to call a function that is not part of its executable code. The executable code for the function is located in a DLL, which contains one or more functions that are compiled, linked, and stored separately from the processes that use them. DLLs also facilitate the sharing of data and resources. Multiple applications can simultaneously access the contents of a single copy of a DLL in memory.

Dynamic linking differs from static linking in that it allows an executable module (either a .DLL or .EXE file) to include only the information needed at run time to locate the executable code for a DLL function. Using DLLs instead of static link libraries makes the size of the executable file smaller. If several applications use the same DLL, this can be a big saving in disk space and memory.

The Advantages of Using DLLs

Dynamic linking has the following advantages:

a. **Saves memory and reduces swapping**. Many processes can use a single DLL simultaneously, sharing a single copy of the DLL in memory. In contrast, Windows must load a copy of the library code into memory for each application that is built with a static link library.

b. **Saves disk space.** Many applications can share a single copy of the DLL on disk. In contrast, each application built with a static link library has the library code linked into its executable image as a separate copy.

c. **Upgrades to the DLL are easy**. When the functions in a DLL change, the applications that use them do not need to be recompiled or relinked as long as the functions' arguments and return values do not change. In contrast, statically linked object code requires that the application be relinked when the functions change.

d. **Provides after-market support.** For example, a display driver DLL can be modified to support a display that was not available when the application was shipped.

e. **Supports multi-language programs**. Programs written in different programming languages can call the same DLL function as long as the programs follow the function's calling convention. The programs and the DLL function must be compatible in the order in which the function expects its arguments to be pushed onto the stack

A potential disadvantage to using DLLs is that the application is not self-contained; it depends on the existence of a separate DLL module.

Chapter 6

Conclusion

The objective of this research was to design a prototype communication system in order to provide a faster and easy to handle information system to retrieve relevant information pertaining to the repair activities. The objective has been accomplished. The prototype is modeled to handle a database relating the activities simulated as per the routine working of base workshop. The simulated repair activity has been tested successfully. Apart from the set objective this project will also provide an easier way for the P&C-section to update and handle the data conveniently. The next section summarizes the methodology of the designing the project and guidelines for the future research.

6.1 Summary and contributions

(This section will be covered in due course)

6.2 Future Research

Research for the designing of this prototype project was aimed to provide a facility to the requesting units as well as to the base workshops to maintain a faster means of communication. TAPI was selected with a view to design such a system. There are still many areas, which could be incorporated in this design work in future. This section outlines some of the issues that must be addressed in order to include various features by future researchers.

The first issue is the continuation of this research and advancing the work of this project. Presently the interface between computer and the caller is through generating DTMF codes i.e. the caller has to press the digits available on the telephone dial pad. This interface could be

improved by introducing another feature known as Speech Application Programming Interface (SAPI). SAPI provides speech recognition features. Speech recognition-Converts audio input into printed text or directly into computer commands. Any speech system has a process for recognizing human speech and turning it into the computer understandable language. In effect, the computer needs a translator. Research into effective speech recognition algorithms and processing models has been going on almost ever since the computer was invented. And a great deal of mathematics and linguistics go into the design and implementation of a speech recognition system. A detailed discussion of speech recognition algorithms is beyond the scope of this section, but it is important to have a brief idea of the technique for turning human speech into computer understandable language.

The reply to the caller by the computer is managed after recording the WAVE FILES in the memory. However this technique has no problem if the recorded files are few but if the number of files are more it is recommended not to over burden the memory. The second important aspect for future research is the inclusion of Text To Speech (TTS) method. TTS, service provides the ability to convert written text into spoken words. There are a number of factors to be considered when developing speech recognition engines (SR). TTS engines use synthesis concatenation technique for turning text input into audio output. Synthesis involves the creation of human speech through the use of stored phonemes. This method results in audio output that is understandable, but not very human-like. The advantage of synthesis systems is that they do not require a great deal of storage space to implement and that they allow for the modification of voice quality through the adjustment of only a few parameters.

6.3 Brief Concept of SAPI and TTS

There are two basic techniques: speech recognition (SR) and speech synthesis, depending on who is doing the talking the person or the computer. Speech synthesis is commonly called "text-to-speech" or TTS, since the speech is usually synthesized from text data. Figure 6.1 shows the architecture of a typical text-to-speech engine.

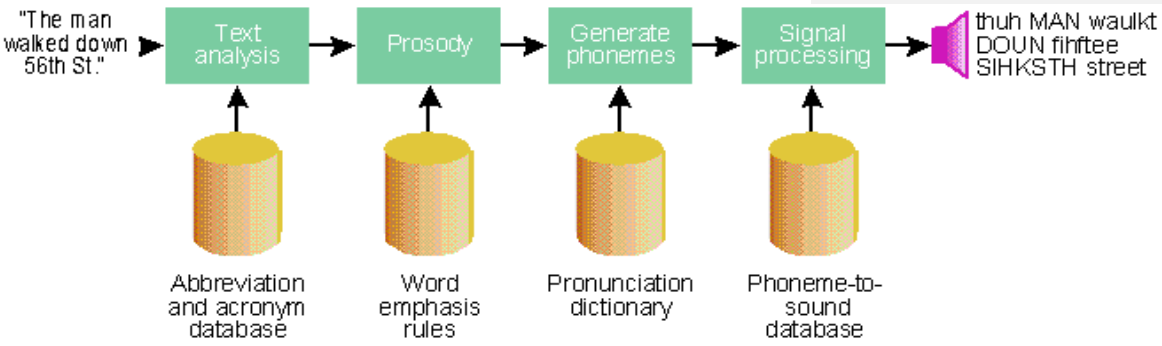

Figure 6.1 Text-to-Speech Engine

The process begins when the application hands over a string of text to the engine a such as, "The man walked down 56th St.". The text analysis module converts numbers into words, identifies punctuation such as commas, periods, and semicolons, converts abbreviations to words, and even figures out how to pronounce acronyms.

Text analysis is quite complex because written language can be so ambiguous. A human has no trouble pronouncing "St. John St." as "Saint John Street," but a computer, in typically mechanical fashion, might come up with "Street John Street" unless a clever programmer gives it some help.

Once the text is converted to words, the engine figures out what words making them louder or longer, should emphasize, or giving them a higher pitch. Other words may be de-emphasized.

Next, the text-to-speech engine determines how the words are pronounced, either by looking them up in a pronunciation dictionary,

or by running an algorithm that guesses the pronunciation. Some text strings have ambiguous pronunciations, such as "read." The engine must use context to disambiguate the pronunciations. The result of this analysis is the original sentence expressed as phonemes. "Th-uh M-A-Nw-au-l-k-tD-OU-Nf-ih-f-t-eeS-IH-K-S-TH s-t-r-ee-t".

Next, the phonemes are parsed and their pronunciations retrieved from a phoneme-to-sound database that numerically describes what the individual phonemes sound like. If speech were simple, this table would have only forty-four entries, one for each of the forty-four English phonemes (or whatever language is used). In practice, each phoneme is modified slightly by its neighbors, so the table often has as many as 1600 or more entries. Depending on the implementation, the table might store either a short wave recording or parameters that describe the mouth and tongue shape. Either way the sound database values are finally smoothed together using signal processing techniques, and the digital audio signal is sent to an output device such as a PC sound card and out the speakers to human’s ears.

Figure 6.2 shows a generic speech recognition engine. When the user speaks, the sound waves are converted into digital audio by the computer's sound card. Typically, the audio is sampled at 11KHz and 16 bits. The frequency analysis module to a more useful format first converts the raw audio. This involves a lot of digital signal processing that's too complicated to describe here. The basic challenge is to extract the meaningful sound information from the raw audio data. If a word is to be said as "foo," and then say "foo" again, and look at the waveforms generated, they would look like similar, but there's no way to compare them that will consistently recognize them as the same sound, without applying some mathematical techniques using Fourier transforms.

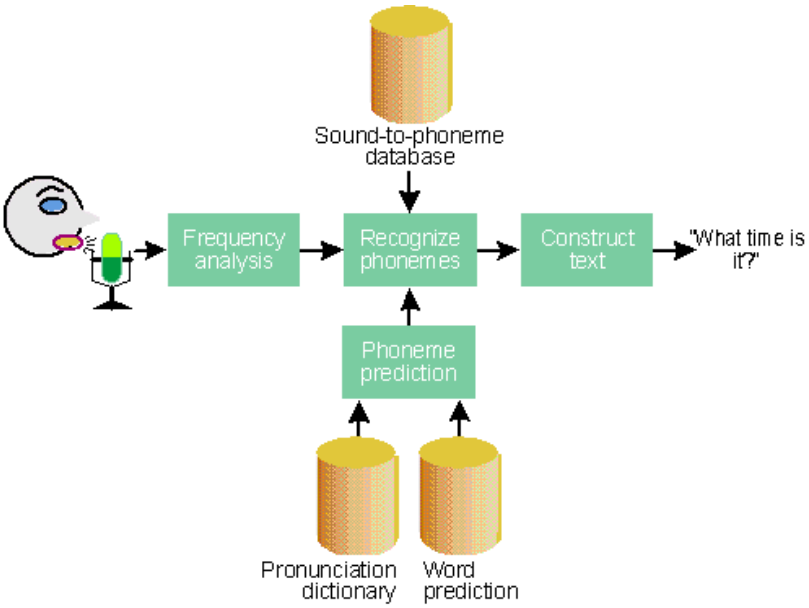

Figure 6.2 Speech Recognition Engine

The converted audio is next broken into phonemes by a phoneme recognition module. This module searches a sound-to-phoneme database for the phoneme that most closely matches the sound it heard. Each database entry contains a template that describes what a particular phoneme sounds like. As with text-to-speech, the table typically has several thousand entries. While the phoneme table could in theory be the same as that used for TTS, in practice they are different because the SR and TTS engines usually come from different vendors.

Because comparing the audio data against several thousand phonemes takes a long time, the speech recognition engine contains a phoneme prediction module that reduces the number of candidates by predicting which phonemes are likely to occur in a particular context. For example, some phonemes rarely occur at the beginning of a word, such as the "ft" sound at the end of the word "raft." Other phonemes never occur in pairs. In English, an "f" sound never occurs before an

"s" sound. But even with these optimizations, speech recognition still takes too long.

A word prediction database is used to further reduce the phoneme candidate list by eliminating phonemes that don't produce valid words. After hearing, "y eh," the recognizer will listen for "s" and "n" since "yes" and "yen" are valid words. It will also listen for "m" in case if it is said "Yemen." It will not listen for "k" since "yek" is not a valid word. (Except in baby-talk, which is not currently supported.) The candidate list can be reduced even further if the application stipulates that it only expects certain words. If the app only wants to know if the user said "yes" or "no," the phoneme recognizer needn't listen for "n" following "y eh," even though "yen" is a word. This final stage reduces computation immensely and makes speech recognition feasible on a 33MHz 486 or equivalent PC. Once the phonemes are recognized, they are parsed into words, converted to text strings, and passed to the application.

As you might imagine, both text-to-speech and speech recognition involve quite a bit of processing, but speech recognition is harder because it usually requires more processing for equivalent user satisfaction. A few years ago, you needed a high-end workstation to do speech recognition. Today, just about every new PC and even many older PCs can handle speech. While the exact requirements vary from one speech engine to another, Figure 3 gives you a rough idea of the hardware needed to run various kinds of speech applications under Windows. The faster the CPU and the more memory available, the higher the accuracy for speech recognition and the better the text-to-speech sounds.

Of course, you also need a sound card, microphone, and speakers. Most speech engines will work with any sound card. Some systems offload processing onto a DSP (digital signal processor) chip

that comes on some high-end sound cards, which cuts the CPU speed requirement in half. Better microphones and speakers will also improve things.

As speech has become more feasible on average PCs, vendors have been busy developing and promoting their speech engines. Many multimedia PCs and sound cards come bundled with speech software. Others vendors sell their engines as standalone products. Some apps even come bundled with speech engines.

Unfortunately, as with any budding technology, the situation is a bit chaotic. Even though they all support similar functionality, each speech engine has its own specific features and proprietary API. If you want to use speech in your app, you've first got to pick which engine to use, and write your program for that engine. If a better engine comes along, you're out of luck. You'll probably have to rewrite your program substantially to use the other API. Proprietary APIs have stifled the widespread adoption of speech. When faced with an irrevocable decision about which engine to use, many developers choose not to implement speech at all.

Chapter's Bibliography

21

Chapter 4

Project Design Details

4.1 Introduction

In the previous chapters it is discussed that what all should be studied before jumping into the design of a conceived project. Having with the under standing of the problem area and acquiring the available solution to handle those problems this chapter in added to discussed in detail the design of the project.

4.2 Database Design Handling

The database is designed under the light of criteria as discussed in chapter 2 under section2.2. This database comprises of a set of tables (relations). Each table has its significance and provides useful information obtain results based on the queries. The significance of each table is discussed in the following paragraphs.

4.2.1 Base Tables.

These tables are designed with a view so that they will provide relevant information during manipulation of data. Tables under this category are discussed in the following section.

4.2.1.1 Table-RegBANo

Registered Broad Arrow Number (RegBANo) table is design to act as the Master record folder that contains all the registration numbers which are under the maintenance load of the workshop are endorsed in this table. BA Number is a unique entity, and no two vehicles exist with identical numbers. The design view of the table is shown as follows.

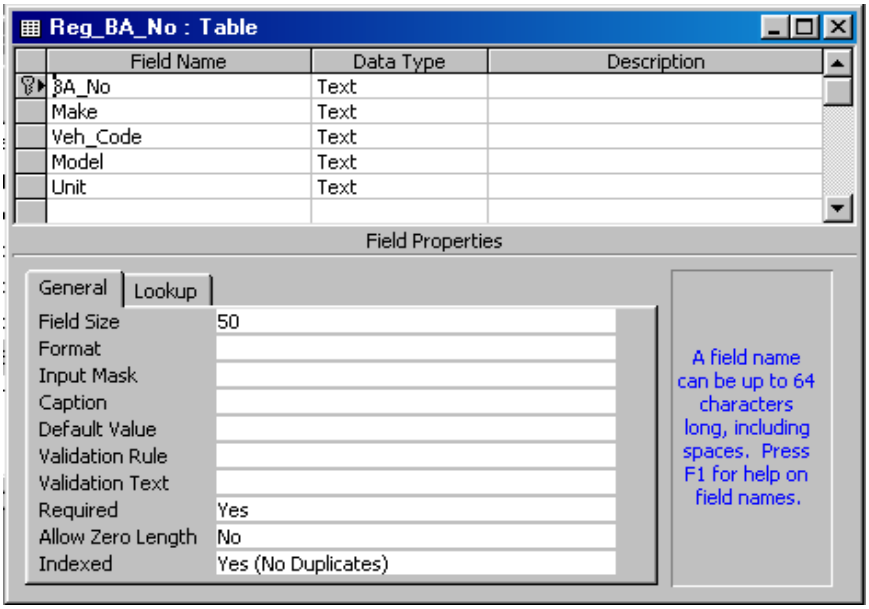

Fig 4.1 Design view of Table RegBANo

The data entered in this table provide the details pertaining to a vehicle such as, the make, model and the unit to which that vehicle belongs are shown in figure 4.2, against a BA number. This information persists till a vehicle exists. The only change in the information appear at the time when either that particular vehicle transferred to some other unit or it is deposited with C.M.T &S.D Golra.

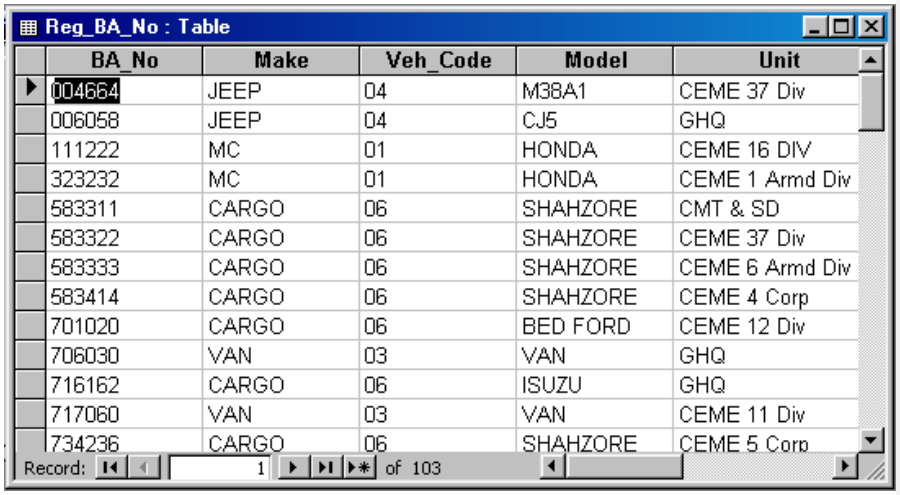

Fig.4.2 Data entries in RegBANo table

4.2.1.2 Project_Code

In order to repair the repairable stock available at C.M.T & S.D Golra, G.H.Q assigns various overhauling project to base workshops. These projects are under taken by the base workshop as a package and each project is assigned a unique project code. For the ease in designing this prototype these project codes are collected in one table. Each project code provides the details that weather the over hauling project is Engine Over hauling Project (E.O.P) or Vehicle Over hauling Project (V.O.P). The situation arises when at a particular time more the one projects of same category are running so for identification these project codes are suffix with serial numbers. The design view and the entries in the table are shown in figure 4.3 and 4.4 respectively.

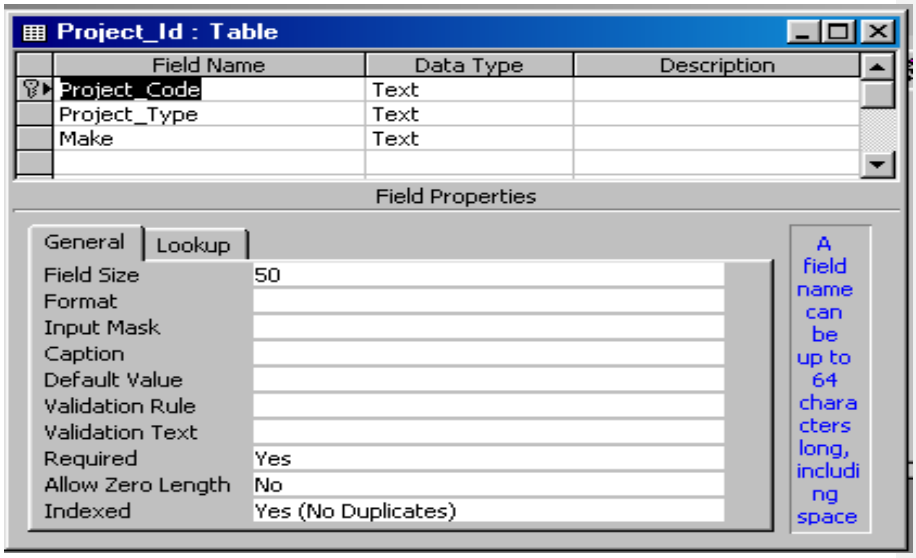

Fig 4.3 Design View of Table project Code

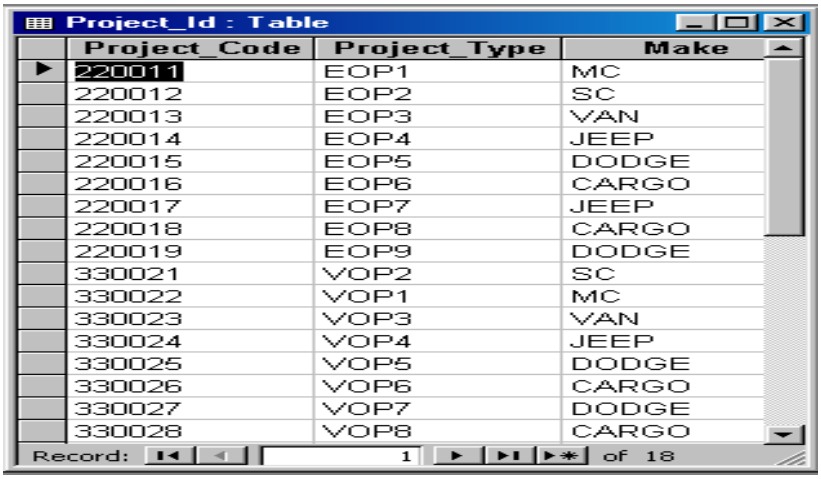

Fig 4.4 Entries in Table Project code.