

REALTIME INTERFACE FOR SIMPUTER

FOR

POWER RESEARCH & DEVELOPMENT CONSULTANTS (P) LTD.,

PROJECT REPORT

P-1241

Submitted in partial fulfillment of the requirements for the award of the
degree of

M.Sc Applied Science Software Engineering

Of Bharathiar University

Coimbatore

Submitted By

PRASHANTH.G

Reg. No. 0137S0045

Guided By

EXTERNAL GUIDE

**Mr.P.Mukund,
Director.**

INTERNAL GUIDE

**Mr.P. Gopalakrishnan, M.C.A,
Lecturer.**



DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641 006

September 2004

CERTIFICATE

This is to certify that the project work entitled

REALTIME INTERFACE FOR SIMPUTER

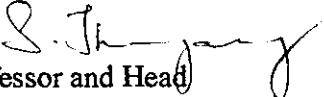
Submitted to the

Department of Computer Science and Engineering

Kumaraguru College of Technology

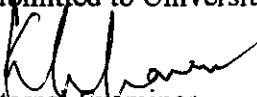


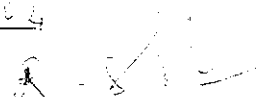
in partial fulfillment of the requirements for the award of the degree of Master of Science [Software Engineering] is a record of original work done by Mr. Prashanth .G, Reg.No. 0137S0045 during his period of study in the Department of Computer Science and Engineering ,Kumaraguru College of Technology, Coimbatore under my supervision and this project work has formed the basis of award of any Degree/Diploma/Associateship/Fellowship or similar title to any candidate of any University.

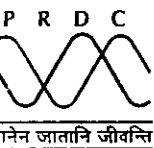

Professor and Head


Staff-in-Charge

Submitted to University Examination held on 30-9-2008


Internal Examiner


External Examiner



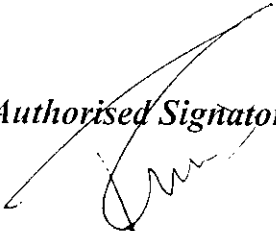
Power Research & Development Consultants Private Limited

#4, I B Main, I N Block,
Rajajinagar, Bangalore - 560 010. INDIA
Tel : +91 (0) 80 2332 2321, 2342 7321
Fax : +91 (0) 80 2352 2917
E-mail : prdc@vsnl.com

CERTIFICATE

This is to certify that the project entitled "REAL TIME INTERFACE FOR SIMPUTER" is submitted to Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore, in partial fulfilment of the requirements of the degree of M.Sc. [Software Engineering], 2001 – 2006 to be awarded by Bharthiar University. The project is a bonafide record work carried out by **Mr. G. Prashanth** with **Reg. No. 0137S0045**, under my supervision and guidance at **Power Research & Development Consultants Pvt. Ltd., Bangalore** during the period June-September 2004.

for POWER RESEARCH & DEVELOPMENT CONSULTANTS PVT. LTD.,

Authorised Signatory


(Mukund Parthasarathy) **Director**
Director **Power Research & Development
Consultants Pvt. Ltd.**

Place: Bangalore

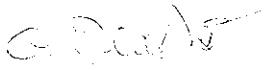
Date: September 9, 2004

DECLARATION

I hereby declare that the project entitled '**REALTIME INTERFACE FOR SIMPUTER**' for Power Research & Development Consultants Private Limited, Bangalore, submitted to Kumaraguru College of Technology, Coimbatore, affiliated to Bharathiar University as the project work of Master of Science[Software Engineering] Degree, is a record of original work done by me under the supervision and guidance of Mr. Mukund .P, MBA and Mr.P.Gopalakrishnan, MCA, and this project work has formed the basis for the award of any Degree/Diploma/Associateship/Fellowship or similar title to any candidate of any University.

Place: Coimbatore

Date: 23-9-2024


Signature of candidate

(G.PRASHANTH)

ACKNOWLEDGMENT

I take this opportunity to express my gratitude to all, whose contribution in this project work can never be forgotten.

I am extremely grateful to **Dr. K.K.Padmanabhan**, Principal, Kumaraguru College of Technology for having given me a golden opportunity to serve the purpose of my education.

I express my sincere thanks to **Prof. Dr. S. Thangaswamy** , B.E(Hon's), Ph.D., Head of the Department of Computer Science and Engineering for making all necessary arrangements for the successful completion of the project.

I owe a great deal to my respected project guide **Mr. P .Gopalakrishnan** , MCA, Lecturer Department of Computer Science and Engineering, for the motivation, constant encouragement and kind suggestion in every step throughout this project.

With immense pleasure, I express my esteemed gratitude to **Dr. R. Nagaraja** , Managing Director, Power Research and Development Consultants Private Limited, Bangalore, for providing me the opportunity to do the project in his esteemed organization.

My sincere gratitude thanks to **Mr. P. Mukund**, Director, Power Research and Development Consultants Private Limited, Bangalore, for his constant help and valuable guidance throughout the project work.

I express my warm thanks and gratitude to my parents, friends and well wishers who have directly and indirectly contributed a lot towards this project.

SYNOPSIS

The primary objective of the project entitled “ **REALTIME INTERFACE FOR SIMPUTER**” is to minimize the quantum of energy loss, which is distributed from the Feeders to various Transformers. The Statistical figure shows that the quantum of energy generated and distributed to the various users, results loss by way of distribution, theft, error in energy meters etc. The quantum of loss is being calculated by using appropriate mathematical formulae.

In order to reduce the energy loss, data is being collected using a Simputer (A hand-held device which is also known as Personal Digital Assistant (PDA) or Simple Computer).

The data is being collected at various points and it's fed into the Simputer. The data from the Simputer is then transferred to the PC via the USB Port. Logistic methodologies are being applied to ascertain the quantum of energy lost.

The platform and language adopted in this project are:

Windows Platform:

- ✓ Front-End: VC++ 6.0
- ✓ Back-End: SQL Server Enterprise Manager

Linux Platform:

- ✓ Front-End: C
- ✓ Back-End: Sqlite

Data present in the PC is used in order to generate reports and analyze the energy losses and thereby submit the status to Bangalore Electricity Supply Company (BESCOM). The concern will take the necessary steps in order to minimize the energy losses. The system also reduces the man-power involved in this work.

CONTENTS

1. INTRODUCTION	
1.1 Project Overview	1
1.2 Organization Profile	2
2. SYSTEM STUDY AND ANALYSIS	
2.1 Existing System – Limitations	4
2.2 User Characteristics	5
2.3 Requirements of the New System	6
2.4 Proposed System	7
3. PROGRAMMING ENVIRONMENT	
3.1 Hardware Configuration	9
3.2 Software Configuration	10
4. SYSTEM DESIGN AND DEVELOPMENT	
4.1 Input Design	27
4.2 Output Design	29
4.3 Process Design	32
4.4 Database Design	37
5. SYSTEM TESTING AND IMPLEMENTATION	
5.1 System Testing	46
5.2 System Implementation	49
6. CONCLUSION	53
7. SCOPE FOR FUTURE DEVELOPMENT	54
8. BIBLIOGRAPHY	55
9. APPENDIX	
9.1 Screen Shots	56

1.INTRODUCTION

1.1 PROJECT OVERVIEW

RealTime Interface for Simputer is a system, which provides better help to users in terms of looking for their beneficiaries. It reduces all the problems involved in the existing system, which may start from collecting the data with pen and paper and then the data will be fed into some worksheet. Most of the calculations will be done manually which may lead to some blunders. It also reduces the cost involved in the current process thus making the system more effective and robust.

Objectives

The objective of the proposed system is to completely reduce the complexities and disadvantages of the current system. The proposed system thereby has increases efficiency and also reduces the cost. It makes data availability and manipulation of data more easy and fast. The system also decreases the large amount of paper work involved in the current process and thus leading to simplicity in the latter stage of process. It also reduces the time consumption of the searching and updating of data involved in the existing system.

The purpose of this system is to eliminate the overhead of the manual process thereby reducing the workload. This system provides maximum user friendliness to the users.

1.2 ORGANIZATION PROFILE

Power Research & Development Consultants Pvt. Ltd. (PRDC) is a dynamic organization with multi-disciplinary experience in providing lucid solutions. PRDC's strong and self-motivated team comprises of highly qualified (Phds, MTechs and B.Es), experienced and dedicated people. Managed by highly qualified technocrats with doctorates from the prestigious Indian Institute Of Science, Bangalore, India.

PRDC owns several products used by a number power utility including many states Electricity Boards and educational institutions across the country. All these products were indigenously designed and developed by PRDC. The most prominent products are:

- ✓ MiPower,
- ✓ PowerDSM
- ✓ MiPADAM
- ✓ PowerEMS and
- ✓ SCADA for power utilities.

PRDC is specialized in developing custom solutions for Automation & Simulation of manufacturing process. It has developed custom applications for the requirements of:

- ✓ Maruti Udyog Limited (India's leading car manufacturers),
- ✓ Hero Honda Motors Limited (India's largest bike manufacturers)
- ✓ TISCO
- ✓ CPRI

- ✓ Bradma Of India Ltd. (leading Automation product suppliers)
to name some.

PRDC also involved in RealT RealTime Embedded Systems Systems. It has lent its services to **ALIND** and **Prok Devices Pvt. Ltd.** Microprocessor based Distance Relay involving DSP was developed for ALIND. Overcurrent & Earth fault Relay (OCEFRelay) was developed indigenously right from preliminary design to prototype and the total technology is transferred to Prok Devices Pvt. Ltd. The mission is to provide stateof-the-art solutions.

PRDC pioneers in lending its services to Power sector in the form of System Planning & Studies. The range of services cater to the gamut of software related activities exclusively for Power sector. We deal with feasibility studies, requirement specifications, hardware configuration and identification, project management, business process, re-engineering, power system tools, custom application development and Real-time embedded systems development. PRDC uses state-of-the-art software and hardware equipment, methodologies (like OOPS) and software development tools.

In its aspiration to design and develop high quality solutions, PRDC follows well-established and field proven standards in all its design and development activities. On the business front, PRDC has strategic associations with Macmet India Limited, Bradma Of India Limited and Advanced Forming Technology Centre (AFTC). Since its inception, PRDC has been steadily advancing towards better-quality, service, innovation and customer satisfaction. PRDC is proud to announce that the company's average growth-rate has touched 60% mark.

2. SYSTEM STUDY AND ANALYSIS

2.1 EXISTING SYSTEM

The existing system is a manual procedure. The person goes to the field and collects the data with the help of a pen and a paper. The data entry person in turn gathers the data and feeds it into a worksheet. Most of the calculations are being done manually, which consumes a lot of man-power, time and cost. The chances of errors are likely to be more in manually work, thereby it lead to malfunctioning of the system. So it consumes a lot of time to debug at which point the error has occurred. Eventually, ultimate goal of the project will not be fulfilled if these types of methodologies are being practiced.

It also may be expensive at times and unaffordable to the beneficiary as the charges of service provided is known only at the latter stages.

Disadvantages

- ✓ Time Consuming
- ✓ Tedious
- ✓ Complex
- ✓ Huge amount of paper work
- ✓ Flow of data is slow
- ✓ Tremendous amount of cost and manpower is involved.

2.2 USER CHARACTERISTICS

The preliminary and crucial stage of the system is the requirement specification and analysis. A deep understanding of the user requirements is essential for the development of a new system because it is ultimately the users who are going to use the system. User involvement in the development of the system is therefore of crucial importance. The greater their involvement, the more effective the system will be.

The project has 2 user levels

- ✓ Field Engineer
- ✓ Administrator

Field Engineer

The Field Engineer is a person who should have a basic knowledge in computers. The field engineers are being trained in order to make them aware of the functionalities of this system. After the training period the engineers are being sent to the field in order to collect the data. This person has only got the privilege to enter the data and to save it.

Administrator

The administrator level has all privileges to the modules in the system. The administrator can export and import the data from device (Simputer). The exporting and importing of data from the device to PC can be controlled in 2 modes namely, TCP/IP and FTP.

Administrator can generate some general and specific reports based upon the need and availability of the data.

2.3 REQUIREMENTS OF THE NEW SYSTEM

In today's emerging new technology we might need to adopt some changes in order for the betterment in our services offered to our clients. As the technology is wide open we must make use of it, for efficient and successful calibration of our goals. Real time Interface for Simputer provides an effective and user-friendly methodology, which understands the longstanding concern, and comes as a solution.

The system will provide

- ✓ The details of all customers under each transformer. The details include the personal details and the energy meter readings for the same.
- ✓ The transfer of data from PC to the device and vice-versa can be done in two ways viz., TCP/IP and FTP.
- ✓ Transfer of data can be done through any one of the technique so that it assures that the will not be lost, thereby time and cost is also contained.
- ✓ The loss of data is very much reduced since the process is automated.
- ✓ User Interface is developed in manner that it's a self-guiding one.
- ✓ The analysis on the data can be made so that the reports can be generated in the logistic manner. This helps out in finding out the specific area where the loss is possible.

2.4 PROPOSED SYSTEM

The proposed system is a self-guiding one. It's being divided in four modules viz.,

- ✓ Meter Readings
- ✓ Transformer Details
- ✓ Customer Details
- ✓ Distribution Overhead Inventory

Meter Readings

Meter Reading module is being used in order to get the monthly reading for the entire customer who fall under that transformer. The field engineers can choose the DTC number and thereby the corresponding RR number is available for selection and finally by selecting the Pole number, the information of that customer can be found on the screen. The engineer can just view the information and exit out of the application or the reading for the customer can also be given in provision.

Transformer Details

Transformer Details module is provided in the sense to add a new transformer or do any changes to the existing transformer.

Customer Details

Customer Details module helps us in order to add a new customer or to do any changes to the existing customers. The possible changes that can be made to the existing customer is like, increasing the number of phase lines etc.

Distribution Overhead Inventory

In this module proper steps are being taken in order to maintain the inventory details.

Advantages

- ✓ Faster information retrieval and updating.
- ✓ Easy Availability of data.
- ✓ Report Generation.
- ✓ Reduces Complexity.
- ✓ Reduces Cost Effectively.
- ✓ Errors are minimized.
- ✓ Time constraints are maintained.

3. PROGRAMMING ENVIRONMENT

3.1 HARDWARE CONFIGURATION

PC:

Processor	Intel Pentium 4 processor
Memory	256 MB
Minimum	256 cache
Display	SVGA Display Adapter
Monitor	SVGA Color Monitor
Keyboard	104 keys
Hard Disk	20 GB
Floppy Drive	1.44 MB
Ports	USB

SIMPUTER:

Processor	Intel StrongArm SA-1110 CPU Battery-backed RTC (Specially designed processor for hand-held devices)
Display	240 x 320 LCD Color with backlight
Memory	32 MB
Hard Disk	64 MB
SmartCard Interface	SmartCard Reader/Writer
Ports	USB

3.2 SOFTWARE CONFIGURATION

PC:

GUI	VC ++ 6.0
Database	SQL Server Enterprise Manager

SIMPUTER:

GUI	Linux C
Database	Sqlite

USER INTERFACE

The Visual C++ user interface consists of an integrated set of windows, tools, menus, toolbars, directories, and other elements that allow you to create, test, and refine your application in one place. You can also work on other types of documents within Visual C++, for example Microsoft Excel or Microsoft Word documents.

The user interface uses standard Windows interface functionality along with a few additional features to make your development environment easy to use. The basic features that you use most often are windows and document views, toolbars, menus, directories, and keyboard shortcuts.

Dynamic Menu Bar

The Menu Bar is context-sensitive. When you begin a new activity, you might notice a new item appearing in the Menu Bar. The drop-down menus listed on the Menu Bar change depending on what you are doing. Some commands are specific to an editor; these are present in a menu only when that editor is in use. Any other command that does not apply to the current situation appears dimmed in the menu.

Shortcut menus

In many of the windows and panes in Visual C++, you can display a shortcut menu by right clicking either a selected item in a list or the background of the pane or window.

The shortcut menu includes common commands related to the area or item. Many of the commands on the shortcut menus are also available by using the Menu Bar. However, you can often issue a command with fewer steps by using the shortcut menu.

Quick access to common dialog boxes

In windows where lists appear, you can often display a commonly used dialog box by double-clicking an item in the list. The particular dialog box that appears depends on the item you double-click and on the context.

Editable property pages

You can display properties by selecting an item, right-clicking it, and clicking Properties in the shortcut menu. The properties are displayed in a property page, which may have several tabs. You can keep this property page on top of other windows by clicking the push-pin button in the upper-left corner. If

any properties are editable, you can change them by editing within the property page.

Context-sensitive toolbar display

When you begin editing or start a new activity, the appropriate toolbars appear for the work you are doing. To display any other available toolbar, click **Customize** on the **Tools** menu, click the **Toolbars** tab, and select the appropriate check box. Many toolbar buttons are equivalent to menu commands, but you can also have toolbar buttons without a counterpart on any menu.

Context-sensitive keyboard shortcuts

Shortcut keys change dynamically, depending on what you are doing. You can display a list of current keyboard shortcuts and assign different shortcut keys.

Quick information about toolbar buttons and commands

You can display quick information about each toolbar button by placing the mouse pointer on the toolbar button. (You do not click or press a mouse button.) A tool tip displays the name of the toolbar button, and a brief description appears in the lower-left corner of the application window (the status bar).

To display a description of the action associated with a menu command, place the mouse pointer over the menu command. The description appears in the status bar.

Customizable toolbars and menus

We can change existing toolbars, create custom toolbars, add menus to toolbars, move or copy toolbar buttons and menu commands, rearrange menus, and change the appearance of buttons and commands.

CSocket

Class **CSocket** derives from **CAsyncSocket** and inherits its encapsulation of the Windows Sockets API. A **CSocket** object represents a higher level of abstraction of the Windows Sockets API than that of a **CAsyncSocket** object. **CSocket** works with classes **CSocketFile** and **CArchive** to manage the sending and receiving of data.

A **CSocket** object also provides blocking, which is essential to the synchronous operation of **CArchive**. Blocking functions, such as **Receive**, **Send**, **ReceiveFrom**, **SendTo**, and **Accept** (all inherited from **CAsyncSocket**), do not return a **WSAEWOULDBLOCK** error in **CSocket**. Instead, these functions wait until the operation completes. Additionally, the original call will terminate with the error **WSAEINTR** if **CancelBlockingCall** is called while one of these functions is blocking.

To use a **CSocket** object, call the constructor, then call **Create** to create the underlying **SOCKET** handle (type **SOCKET**). The default parameters of **Create** create a stream socket, but if you are not using the socket with a **CArchive** object, you can specify a parameter to create a datagram socket instead, or bind to a specific port to create a server socket. Connect to a client socket using **Connect** on the client side and **Accept** on the server side. Then create a **CSocketFile** object and associate it to the **CSocket** object in the **CSocketFile** constructor. Next, create a **CArchive** object for sending and one for receiving data (as needed), then associate them with the **CSocketFile** object in the **CArchive** constructor. When communications are complete, destroy the **CArchive**, **CSocketFile**, and **CSocket** objects.

Datagram Sockets

Datagram sockets support a bi-directional data flow that is not guaranteed to be sequenced or unduplicated. Datagrams also are not guaranteed to be reliable; they can fail to arrive. Datagram data may arrive out of order and possibly duplicated, but record boundaries in the data are preserved, as long as the records are smaller than the receiver's internal size limit. You are responsible for managing sequencing and reliability. (Reliability tends to be good on local area networks (LANs) but less so on wide area networks (WANs), such as the Internet.)

Datagrams are "connectionless" - no explicit connection is established; you send a datagram message to a specified socket and you can receive messages from a specified socket.

An example of a datagram socket is an application that keeps system clocks on the network synchronized. This illustrates an additional capability of datagram sockets in at least some settings: broadcasting messages to a large number of network addresses. Datagram sockets are better than stream sockets for record-oriented data.

Stream Sockets

Stream sockets provide for a data flow without record boundaries - a stream of bytes that can be bi-directional (the application is full-duplex: it can both transmit and receive through the socket). Streams can be relied upon to deliver sequenced, unduplicated data. ("Sequenced" means that packets are delivered in the order sent. "Unduplicated" means that you get a particular packet only once.) Receipt of stream messages is guaranteed, and streams are well-suited to handling large amounts of data.

The network transport layer may break up or group data into packets of reasonable size. The **CSocket** class will handle the packing and unpacking for you. Streams are based on explicit connections: socket A requests a connection to socket B; socket B accepts or rejects the connection request.

A telephone call provides a good analogy for a stream: under normal circumstances, the receiving party hears what you say in the order that you say it, without duplication or loss. Stream sockets are appropriate, for example, for implementations such as the File Transfer Protocol (FTP), which facilitates transferring ASCII or binary files of arbitrary size.

Stream sockets are preferable to datagram sockets when the data must be guaranteed to arrive and when data size is large.

Externally Creatable Objects

Most large ActiveX-enabled applications and other ActiveX components provide a top-level externally creatable object in their object hierarchy that:

- ✓ Provides access to other objects in the hierarchy.
- ✓ Provides methods and properties that affect the entire application.

In addition to these top-level externally creatable objects, ActiveX components can also provide externally creatable objects that are lower on the component's object hierarchy. You can access these objects either directly as an externally creatable object or indirectly as a dependent object of a higher-level externally creatable object.

Dependent Objects

A reference to a dependent object in only one way - by using a property or method of an externally creatable object to return a reference to the dependent object. Dependent objects are lower in an object hierarchy, and only using a method of an externally creatable object can access them.

Code reuse

Using the mechanisms I just described, you can create your own classes that offer services to third-party applications. This, of course, makes it extremely simple to reuse code and broadens the market for the component vendors.

SQL Server Enterprise Manager

SQL Server Enterprise Manager is a graphical tool that allows for easy, enterprise-wide configuration and management SQL Server and SQL Server objects. SQL Server Enterprise Manager provides:

- ✓ A scheduling engine
- ✓ Administrator alert capability.
- ✓ Drag-and-drop control operations across multiple servers
- ✓ A built-in replications management interface.

We can also use SQL Server Enterprise Manager to:

- ✓ Manage logins, permissions and users.
- ✓ Create scripts.
- ✓ Manage devices and databases.
- ✓ Backup databases and transaction logs.

- ✓ Manage tables, views, stored procedures, triggers, indexes, rules, defaults, and user-defined data types.

Creating and Maintaining Databases

Designing the Microsoft SQL Server database structure involves creating and maintaining a number of interrelated components.

Database Component	Description
Database	Contains the objects used to represent, manage and access data.
Tables	Store rows of data and define the relationships between multiple tables.
Database Diagrams	Represents database objects graphically and enables to interact with the database without using Transact-SQL.
Indexes	Optimize the speed of accessing the data in the table.
Views	Provide an alternate way of looking at the data in one or more tables.
Stored Procedures	Centralize business rules, tasks and process within the server using Transact-SQL Programs.
Triggers	Centralize business rules, tasks and process within the server using special types of stored procedures that are only executed when data in a table is modified.

Accessing and Changing Data

SQL Server Enterprise Manager includes a tool for designing queries interactively using a Graphical User Interface (GUI). These queries are used:

- ✓ In views.
- ✓ In Data Transformations Services (DTS) Packages.
- ✓ To display the data in Microsoft SQL Server tables.

Replication

Replication is an important and powerful technology for distributing data and stored procedures across an enterprise. The replication technology SQL Server allows you to make copies of your data, move those copies to different locations, and synchronize the data automatically so that all copies have the same data values. Replication can be implemented between databases on the same server or different servers connected by LANs, WANs, or the Internet. The procedures in this section help to configure and maintain replications using SQL Server Enterprise Manager.

Data Transformation Services

Data Transformation Services (DTS) provides the functionality to import, export and transform data using COM, OLE DB and Microsoft ActiveX Scripts. DTS enables to build and manage data marts and data warehouses by providing:

- ✓ An extensible transaction-oriented workflow engine that allows execution of a complex series of operations.
- ✓ Powerful integrated heterogeneous data movement, scrubbing, and movement. DTS can copy, validate and transform data from many popular desktop and server-based data sources including Microsoft Access, dBase, Microsoft Excel, Microsoft Visual FoxPro, Paradox, SQL Server, Oracle and DB2.

- ✓ An industry standard method of sharing metadata and data lineage information through Microsoft Repository. Leading data warehousing and database design vendors have adopted this information model.
- ✓ Package storage in Microsoft Repository, SQL Server or COM-structured storage files. After a package has been saved, it can be scheduled for execution using a SQL Server Agent.
- ✓ Extensibility that allows advanced users to meet their unique needs while continuing to leverage DTS functionality.
- ✓ Integration with Microsoft SQL Sever OLAP Services.

Managing Security

To ensure that only authorized users access data and objects stored in Microsoft SQL Server, security must be set up correctly. Understanding how to set up security correctly can help ongoing management. Security elements that may have to be set up include authentication models, logins, users, roles, granting, revoking, and denying permissions on Transact-SQL statements and objects, and data encryption.

Databases

A database in Microsoft SQL Server consists of a collection of tables with data, and other objects, such views, indexes, stored procedures, and triggers that are defined to support the activities performed with data. Before objects within the database can be created, we must create the database and understand how to change the settings and configuration of the database. This includes tasks such as expanding or shrinking the database, or specifying the files used to create the database.

Tables

Tables are database objects that contain all the data in the database. A table definition is a collection of columns in the same way a database is a collection of tables. Before data can be stored in a database we must understand how to create, modify, and maintain the tables within the database. This includes tasks such as defining keys and adding or deleting columns from a table.

Database Diagrams

Database diagrams enable us to create, manage, and view database objects in a graphical format. Before objects within the database can be manipulated using database diagrams, we must understand how to create a database diagram, add objects to it, work within a database diagram and a save a database diagram.

Indexes

To create efficient indexes that improve the performance of the database applications by increasing the speed of queries, we need an understanding of how to create and maintain the indexes on the tables in our database.

Views

By creating, modifying and maintaining views, we can customize each user's perception of the database.

Stored Procedures

By creating, modifying and using stored procedures, we can simplify the business applications and improve application and database performance.

Triggers

By understanding how to create, modify and maintain triggers, we can use triggers to:

- ✓ Cascade changes through related tables in the database.
- ✓ Disallow or rollback changes that violate referential integrity, thereby canceling the attempted data modification transaction.
- ✓ Enforce restrictions that are more complex than those defines with CHECK constraints.
- ✓ Find the difference between the state of a table before and after a data modification and take action(s) based on that difference.

SQLite -An Embeddable SQL Database Engine

Introduction

SQLite is a C library that implements an embeddable SQL database engine. Programs that link with the SQLite library can have SQL database access without running a separate RDBMS process. The distribution comes with a standalone command-line access program (sqlite) that can be used to administer an SQLite database and which serves as an example of how to use the SQLite library.

SQLite is not a client library used to connect to a big database server. SQLite is the server. The SQLite library reads and writes directly to and from the database files on disk.

Features

- ✓ Implements most of SQL92.
- ✓ A complete database (with multiple tables and indices) is stored in a single disk file.

- ✓ ACID (Atomic, Consistent, Isolated, Durable) transactions.
- ✓ Database files can be freely shared between machines with different byte orders.
- ✓ Supports databases up to 2 terabytes (2^{41} bytes) in size.
- ✓ Small memory footprint: less than 25K lines of C code.
- ✓ Two times faster than PostgreSQL and MySQL for many common operations.
- ✓ Very simple C/C++ interface requires the use of only three functions and one opaque structure.
- ✓ TCL bindings included. Bindings for many other languages available separately.
- ✓ Simple, well commented source code.
- ✓ Automated test suite provides over 90% code coverage.
- ✓ Self-contained: no external dependencies.
- ✓ Built and tested under Linux and Windows.

Cost Estimation and Scheduling

Software cost is related to many variables- Human, Technical, Environment, Political and effort applied to develop it. However, software project estimation can be transformed from a blank art to serried of systematic steps that provide estimates with acceptable risks.

The estimates of cost depend, in turn, on our ability to estimate and evaluate several factors, given below:

- ✓ Experience and ability of the proposal personal.
- ✓ The quality of software development environment.
- ✓ The degree to which our understanding of the problem and its acceptable solutions is likely to change.

- ✓ The complexity of the eventual code.
- ✓ The degree to which software components can be reused.
- ✓ The degree of market readiness for the product.
- ✓ The amount of evaluation the product will eventually undergo.

The cost estimation of the proposed is purely based on the requirements of the manpower and consumptions of time involved in finishing the proposed system. Cost estimation also includes hardware and software, which is used for developing the proposed system. Cost estimation identifies the value of the proposed system. Cost estimation and scheduling is done before and after the development of the proposed system.

4. SYSTEM DESIGN AND DEVELOPMENT PROCESS

Design is a creative process; a good design is the key to effective system. The term “Design” is defined as “The process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realizations”. Various design features are followed to develop the system. The design specification describes the features of the system, the components or elements of the system and their appearance to end-users.

In a system design high-end decisions are taken regarding the basic system architecture, platforms and tools to be used. The system design transforms a logical representation of what a given system is required to be in to the physical specifications. Design starts with the system’s requirement specifications and converts it into a physical reality during the development. Important design factors such as reliability, response, time, throughput of the system, maintainability, expandability, etc, should be taken into account.

Fundamental Design Concepts

A set of fundamental design concepts is evolved over the past three decades. Although the degree of interest in each concept has varied over the years, each has stood the test of time. Each provides the software designer with a foundation from which more sophisticated design methods can be applied. Fundamental design concepts provide the necessary framework for “getting it right”.

Abstraction

Abstraction permits one to concentrate on a problem at some level of generalization without regard to irrelevant low-level details, use of abstraction also permits one to work with concepts and terms that are familiar in the problem environment without having to transform them to an unfamiliar structure. Two types of abstraction are:

- ✓ Procedural Abstraction
- ✓ Data Abstraction

A procedural abstraction is a named sequence of instructions that has a specific and limited function. A data abstraction is named collection of data that describes a data object.

Modularity

Modularity is the single attribute software that allows a program to be intellectually manageable. Software architecture embodies modularity, that is, software is divided into named and addressable components, called modules that are integrated to satisfy problem requirements.

Software Architecture

Software Architecture alludes to “the overall structure of the software and the ways in which that structure provides conceptual integrity for a system”. Control hierarchy also called program structure, represents the organization of control. The tree structure used to represent the control hierarchy.

Structural Partitioning

The program structure should be partitioned both horizontally and vertically. Horizontal partitioning defines separate branches of the modular hierarchy for each major program function. Vertical partitioning called factoring, suggest that control and work should be distributes top-down in the program architecture. Top-level modules should perform control functions and do little actual processing work. Modules reside low in the architecture should be the workers, performing all input, computational, an output task.

Data Structure

Data structure is a representation of logical relationship among individual elements of data. Because the structure of information will invariably affects the final procedural design, data structure is very important as the program structure to representation of the software architecture. Data structure dictates the organization, methods of access, degree of associatively, and processing alternatives for information. The organization and complexity of a data structure are limited only by the ingenuity of the designer. Scalar item array and linked list are some of the representations of the data structure.

Software Procedure

Program structure defines control hierarchy without regard to the sequence of processing and decisions. Software procedure focuses on the processing details of each module individually. Procedure must provide a precise specifications of processing, including sequence of events, exact, decision points, respective operations and even data organization/structure. Information hiding suggests that modules be “characterized by design decisions that hide from all others.” In other words, modules should be specified and designed so that information contained within module is inaccessible to other modules.

4.1 INPUT DESIGN

Input design is the link between the information system and the users and those steps that are necessary to put transaction data into a usable form for processing data entry. Instructing the computer to read data from a written printed document can activate the activity of putting data into the computer for processing or it can occur by keeping data directly into the system. The designs of input focusing on controlling the amount of input required controlling the errors, avoid delay extra steps, and keeping the process simple. System analyst decides the following input design details:

- a. What data to input?
- b. What medium to use?
- c. How the data is arranged and coded?
- d. The dialogue to guide the users in providing input.
- e. Data items and transaction needing validation to detect errors.
- f. Methods for performing input validation and steps to follow when error occurs.

The major input forms are:

PC:

- ✓ Export Form
- ✓ Import Form
- ✓ Report Simulation Form

Simputer:

- ✓ Meter Reading Form
- ✓ Consumer Installation Details Form
- ✓ Distribution OverHead Inventory Form
- ✓ Transformer Details Form

Export Form

The specific transformers are being selected and its being send to the simputer from the PC via the USB port.

Import Form

In the case of importing we make selections between the forms (Meter Reading Form, Consumer Installation Details Form, Distribution OverHead Inventory Form, Transformer Details Form) and then transfer it from the simputer to the PC via the USB port

Report Simulation Form

The reports are generated for the selected transformer for the specific interval.

Meter Reading Form

The quantum of energy consumed by each consumer is being collected in the filed using this form.

Consumer Installation Details Form

The details of a new consumer are being added to this form.

Distribution OverHead Inventory Form

The inventory detail for each consumer is being metered by using this form, in order to estimate the loss.

Transformer Details Form

In case, if a new transformer is being added then the complete details are being entered in this form.

4.2 OUTPUT DESIGN

Designing computer should proceed in well thought out manner. The term output means any information produced by the information system whether printed or displayed. When analyst design computer output they identify the specific output that is needed to meet the requirement. Computer is the most important source of information to the users. Output design is a process that involves designing necessary outputs that have to be used by various users according to requirements. Efficient intelligent output design should improve the system relationship with the user and help in decision making. Since the management for taking decisions and to draw the conclusion directly requires the reports must be simple, descriptive and clear to the user. Options for outputs and forms are given in the system menus.

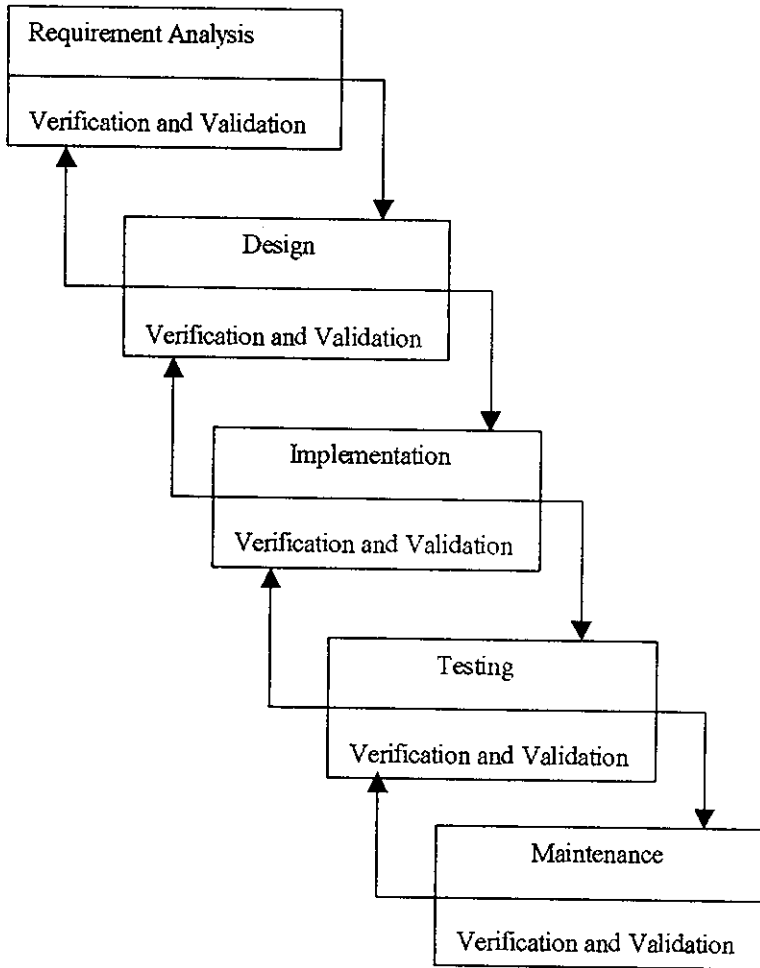
When designing the output, system analyst must accomplish the following:

- a. Determine the information to present.
- b. Decide whether to display, print, speak the information and select the output medium.

- c. Arrange the information in acceptable format.
- d. Decide how to distribute the output to intended receipt.

Development Approach

Integrated Software System for Enterprise Resource Scheduling was designed and developed based on the Waterfall Model. This model particularly expresses the interaction between subsequent phases. Testing software is not an activity, which strictly follows the implementation phase. In each phase of the software development process, we have to compare the results obtained against that which is required. In all quality has to be assessed and controlled.



Various Phases in System's Life Cycle

4.3 PROCESS DESIGN

Design Notations

Design defining a model of the new system and continues by converting this model to a new system. There are methods used to convert the model of the proposed system into computer specifications. Data models are converted to a database and processes and flows to user procedures and computer programs. Design proposes the new system that meets these requirements. This new system may be built a fresh or changing the existing system. The detailed design starts with three activities viz.,

- ✓ Database Design
- ✓ User Interface Design

Database design uses conceptual data model to produce a database design.

User design uses those parts of the DFD outside the automation boundary to design user procedures.

Data Flow Diagram

The data flow diagram (DFD) is one of the most important tools used by system analysts. Data flow diagrams are made up of a number of symbols, which represent system components. Most data flow modeling methods use four kinds of symbols. These symbols are used to represent four kinds of system components viz., processes, data stores, and external entities. Circles in DFD represent processes.

Data Flow represented by a thin line in the DFD and each data store has a unique name and square or rectangle represents external entities. Unlike detailed flowchart, Data Flow Diagrams do not supply detailed description of the modules but graphically describes a system's data and how the data interact with the system.

To construct a Data Flow Diagram, we use,

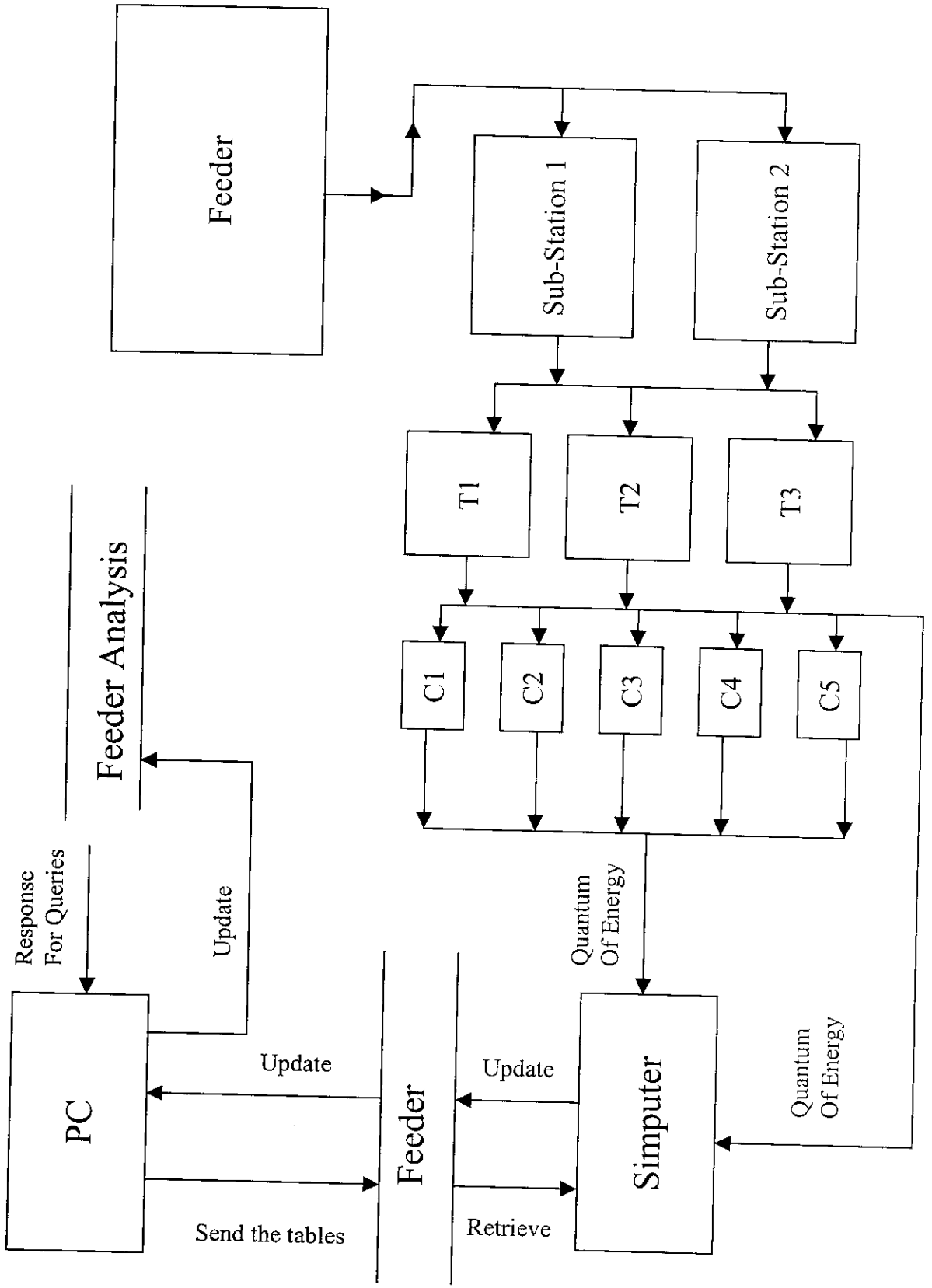
- ✓ Arrow
- ✓ Circles
- ✓ Open End Box
- ✓ Squares

An arrow identifies the data flow in motion. It is a pipeline through which information is flown like the rectangle in the flowchart. A circle stands for process that converts data into information. An open-ended box represents a data store, data at rest or a temporary repository of data. A square defines a source or destination of system data.

Rules for constructing a Data Flow Diagram are:

- ✓ Arrows should not cross each other.
- ✓ Squares, circles and files must bear names.
- ✓ Decomposed data flow squares and circles can have same names.

DFD for Realtime Interface for Simputer



Structured Chart

Structure chart is made up of program modules and the interconnection between them. A rectangular box in the structure chart represents this program module. Modules at the top level of the structure chart call the modules at the lower levels. Lines between the rectangular boxes represent the connection between modules. The connection describes data flow between the called and calling modules.

As well as a DFD, it is also useful to develop a structural system mode. This structural model shows how a function is realized by a number of other functions, which it calls. Structure charts are a graphical way to represent this decomposition hierarchy. Like DFD, they are dynamic rather than static system models. They show how one function calls others. They do not show a static block structure of a function or procedure.

A function is represented on a structure chart as a rectangle. The hierarchy is displayed by linking rectangles with lines. Inputs and outputs are indicated with annotated arrows. An arrow entering a box implies input, leaving a box implies output. Data stores are shown as rounded rectangles and user inputs as circles.

Rules

- ✓ Many systems can be considered as three stages viz., input, validation and output.
- ✓ If data validation is required, function to implement these should be subordinate to an input function.

- ✓ The role of function near the top of the structural hierarchy may be to control and coordinate a set of lower level hierarchy.
- ✓ The objective of design process is to have loosely coupled highly cohesive components.
- ✓ Each node in the structure chart should have between two and seven subordinates.

ER Diagram

A conceptual model describes the essential feature of system data. This conceptual model is described by modeling method known as Entity-Relationship analysis. Entity-Relationship analysis uses three major abstractions to describe data. These are entities- which are distinct things in the enterprise. Relationship-which are meaningful interactions between the objects and the attributes- which are properties of entities and relationship.

To construct an ERD, we use

- ✓ Rectangle
- ✓ Ellipses
- ✓ Diamond

4.4 DATABASE DESIGN

Design begins when management approves the feasibility study produced during detailed analysis and authorizes the necessary funds and personnel to continue. It concludes when management approves the design and authorizes development of the actual system.

A database is a collection of interrelated data stored with minimum redundancy to server many users quickly and efficiently. The general objective of database design is to make the data access easy, inexpensive and flexible to the user.

Rules of Data Normalization

- ✓ Make a separate table for each set of related attributed, and give each table a primary key.
- ✓ **Eliminate Redundant Data:**
If an attribute depends on only part of a multi-valued key, remove it to a separate table.
- ✓ **Eliminate Columns Not Dependant On Key:**
If attributes do not contribute to a description of the key, remove them to a separate table.
- ✓ **Isolate Independent Multiple Relationships:**
No table may contain two or more 1:n or n:m relationships that are not directly related.
- ✓ **Isolate Semantically Related Multiple Relationships:**
There may be practical constrains on information that justify separating logically related Many-to-Many relationships.

✓ **Domain-Key Normal Form:**

A model free from all modification anomalies.

✓ **Optimal Normal Form:**

A model limited to only simple (elemental) facts, as expressed in ORM.

TABLE DESIGN

Table: ConsumerInstallationDetails

Sl. No	Field Name	Data Type	Constraint	Description
1	szDTCNo	Varchar2(50)	Not Null	Transformer no. of Consumer
2	szPoleNo	Varchar2(50)	Not Null	Pole no. of Consumer
3	iSINo	Integer		Serial no.
4	szName	Varchar2(50)		Name of Consumer
5	szAddress	Varchar2(50)		Address of Consumer
6	szPhoneNo	Varchar2(50)		Phone no. of Consumer
7	szRRNo	Varchar2(50)	Primary Key	RR no. of Consumer
8	iReadingDate	Integer		Monthly Reading date of Consumer
9	szLDNo	Varchar2(50)		LD no.of Consumer
10	szLFNo	Varchar2(50)		LF no. of Consumer
11	szTariff	Varchar2(50)		Tarrif of of Consumer
12	iCategoryInstallation	Integer		Category of Consumer
13	dTotalConnectedLoad	Integer		Max. Load of Consumer
14	szMake	Varchar2(50)		Name of the concern manufacturing Meters
15	szModel	Varchar2(50)		Model of energy meter
16	iMeterType	Integer		Type of energy meter
17	szMeterNo	Varchar2(50)		Serial no.of energy meter
18	szPONoYear	Varchar2(50)		Year of Manufacturing
19	iPhaseType	Integer		Phase type of Consumer
20	iEarthingType	Integer		Earthing type of Consumer
21	dCurrentMeterReading	Integer		Current Meter Reading of Consumer
22	dtDateVisit	Date/Time		Date of visit
23	iMainCoverSealStatus	Integer		Status of MainCoverSeal
24	iTerminalCoverSealStatu	Integer		Status of TerminalCoverSeal
25	iMCCBPresent	Integer		Status of Moulded Case Circuit Breaker
26	dRevPerPulse	Integer		Revolutions per pulse of the energy meter
27	dAmp	Varchar2(50)		Amp of Energy Meter
28	dRev	Varchar2(50)		Revolutions
29	dTime	Varchar2(50)		Time Consumed
30	iMeterCondition	Integer		Condition of the energy meter
31	szRemarks	Varchar2(50)		Comments

Table: DistributionOverHeadInventory

Sl. No	Name	Data Type	Constraint	Description
1	DTCNo	Varchar2(50)	Not Null	Transformer no. of Consumer
2	PoleNo	Varchar2(50)	Not Null	Pole no. of Consumer
3	iPoleType	Integer		Type of the Pole
4	dSpanLength	Integer		Distance between two poles
5	dRabbit	Integer		Commerical name of conductor
6	dSquirrel	Integer		Commerical name of conductor
7	dWeasel	Integer		Commerical name of conductor
8	i4pincross	Integer		4 Pin cross arm(Low Tension)
9	i2pincross	Integer		2 Pin cross arm(Low Tension)
10	istsupport	Integer		Description of stsupport
11	i11kV3pincross	Integer		Description of 11kV 3 Pin cross arm
12	i11kv2pincross	Integer		Description of 11kV 2 Pin cross arm
13	i11kVVcross	Integer		Description of 11 kV VV Pin cross arm
14	isignletlf	Integer		No. of Single tubelight fitting
15	idoubletlf	Integer		No. of Double tubelight fitting
16	isvlamp	Integer		No. of Soudium Vapour lamps
17	imvlamp	Integer		No. of Mercury Vapour lamps
18	ignfitting	Integer		Description of gn fittings
19	ihmlamp	Integer		No. of hm lamps
20	iguy	Integer		No. of guy sets
21	istudpole	Integer		No. of stud poles
22	ipininsulator	Integer		No.of pin insulators
23	istraininsulator	Integer		No. of strain insulators (High and Low Tension)
24	ishakleinsulator	Integer		No. of shakle insulators (High Tension)
25	ipininsulator11kv	Integer		No. of 11 kV pin insulators
26	istraininsulator15kv	Integer		No. of 15 kV strain insulators
27	idiscinsulator	Integer		No. of Disc insulators (High and Low Tension)
28	isphservice	Integer		Single phase service
29	i3phservice	Integer		Three phase service
30	iStreetLightControlSwitc	Integer		Condition of Street light control switch
31	i3ph4wire	Integer		Condition of three phase 4 wire
32	i3ph5wire	Integer		Condition of three phase 5 wire
33	isph2wire	Integer		Conditon of single phase 2 wire
34	isph3wire	Integer		Conditon of single phase 3 wire
35	iothers	Integer		Value of others
36	szPhysicalConditionPole	Varchar2(50)		Condition of pole
37	szRemarks	Varchar2(50)		Comments

Table: TransformerDetails

Sl. No.	Name	Data Type	Constraint	Description
1	szDivisionName	Varchar2(50)	Not Null	Name of the division
2	szSubDivisionName	Varchar2(50)	Not Null	Name of the sub-division
3	szDTCNo	Varchar2(50)	Not Null	Transformer No.
4	szDTCName	Varchar2(50)		Transformer Name
5	szTransMake	Varchar2(50)		Transformer Make
6	szSlNo	Varchar2(50)		Serial No.
7	dTransformerRating	Integer		Rating of Transformer
8	iVectorGroup	Integer		Range of Vector group
9	dVoltagePrimary	Integer		Total primary voltage
10	dVoltageSecondary	Integer		Total secondary voltage
11	dNoLoadLoss	Integer		Total no load loss
12	dLoadLoss	Integer		Total load loss
13	iManufacturingYear	Integer		Manufacturing year of the transformer
14	i11KVGOS	Integer		No. of 11 kV Gang Operator Switch
15	iHTSideProtection	Integer		Provision for High Tension side protection
16	iLTSideProtection	Integer		Provision for Low Tension side protection
17	iNoLTLines	Integer		No. of Low Tension Lines
18	iOilLevel	Integer		Oil level of transformer
19	iEarthingCondition	Integer		Earthing Conditon of transformer
20	iTransformerCondition	Integer		Condition of transformer
21	dtETVMFixingDate	Date/Time		Fixing date of Volt Meter
22	szManufacturer	Varchar2(50)		Name of manufacturer
23	iETVMSiNo	Integer		Serial No. of Volt Meter
24	dAccuracyClass	Integer		Class of accuracy of transformer
25	dMeterConstPrimary	Integer		Value of primary meter constant
26	dMeterConstSecondary	Integer		Value of secondary meter constant
27	dtTestedDate	Date/Time		Tested date of the transformer
28	dinitReading	Integer		Initial reading of the transformer
29	szCTMake	Varchar2(50)		Make of the Current Transformer
30	szCTSiNo	Varchar2(50)		Serial no. of the Current Transformer
31	dCTAccuracyClass	Integer		Class accuracy of the Current Transformer
32	szPolarityMarking	Varchar2(50)		Remarks on polarity marking
33	szCTTestingDatePlace	Date/Time		Tested date and place
34	szComments	Varchar2(50)		Remarks

Table: MeterReading

Sl. No.	Name	Data Type	Constraint	Description
1	szRRNoRDTCNo	Varchar2(50)	Not Null	Description of transformer no. or RR no.
2	szDTCNo	Varchar2(50)	Not Null	Description of transformer no.
3	dtReadingDate	Date/Time		Reading date of each consumer
4	fMeterReading	Integer		Meter Reading of each consumer
5	fMeterConstant	Integer		Meter Constant of each consumer

Table: TransformerDailyReadings

Sl. No.	Name	Data Type	Constraint	Description
1	iUnitCode	Integer		Unit Code
2	dtDateTime	Date/Time	Not Null	Date and time of visit
3	dForKVA	Integer		Kilo Voltage Amp- Apparent Power
4	dForKW	Integer		Kilo Watt-Real Power
5	dForKVAR+	Integer		Kilo Voltage Amp - Reactive Power(Leading)
6	dForKVAR-	Integer		Kilo Voltage Amp - Reactive Power(Lacking)
7	dRphCurrent	Integer		Red Phase Current
8	dYphCurrent	Integer		Yellow Phase Current
9	dBphCurrent	Integer		Blue Phase Current
10	dRphVoltage	Integer		Red Phase Voltage
11	dYphVoltage	Integer		Yellow Phase Voltage
12	dBphVoltage	Integer		Blue Phase Voltage

Table: ConsumerCategoryReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	iCategoryNo	Integer		Category Number
2	szCategoryType	Varchar2(50)		Category Type

Table: EarthingConditionReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	iEarthingCondition	Integer		Condition of Earthing
2	szDescription	Varchar2(50)		Comments

Table: HTProtectionReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	iHTProtectionType	Integer		Description of High Tension Protection Type
2	szProtectionDescri	Varchar2(50)		Comments

Table: LoadNoLoadLossReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	dKVARating	Integer		Kilo Voltage Amp.
2	dNoLoadLoss	Integer		Value of NoLoad Loss
3	dLoadLoss	Integer		Value of Load Loss

Table: LTProtectionReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	iLTProtectionType	Integer		Low Tension Protection Type
2	szProtectionDescri	Varchar2(50)		Comments

Table: MakeReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	iMake	Integer		Make of Transformer
2	szDescription	Varchar2(50)		Comments

Table: MeterTypeReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	iType	Integer		Type of Meter
2	szDescription	Varchar2(50)		Comments

Table: OilLevelReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	iOilLevelType	Integer		Type of Oil Level in Transformer
2	szDescription	Varchar2(50)		Comments

Table: PoleTypeReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	iType	Integer		Type of pole
2	szPoleDescription	Varchar2(50)		Comments

Table: TariffReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	iTariffType	Integer		Type of Tariff
2	szDescription	Varchar2(50)		Comments

Table: TransformerConditionReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	iTransformerCondi	Integer		Condition of Transformer
2	szDescription	Varchar2(50)		Comments

Table: VectorGroupReferenceTable

Sl. No.	Name	Data Type	Constraint	Description
1	iVectorGroupID	Integer		Notation of Vector
2	szTypeDescription	Varchar2(50)		Comments

5. SYSTEM TESTING AND IMPLEMENTATION

5.1 SYSTEM TESTING

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated and perform allocated functions.

Testing is the final verification and validation activity within the organization itself. In the testing stage, efforts we put on to achieve the following goals:

- ✓ To affirm the quality of the product.
- ✓ To find and eliminate any residual errors from previous stages.
- ✓ To validate the software as a solution to the original problem.
- ✓ To demonstrate the presence of all specified functionality in the product.
- ✓ To estimate the operational reliability of the system.

During testing the major activities are concerned on the examination and modification of the source code.

Testing Methodologies

The following are testing methodologies:

- ✓ Unit Testing.
- ✓ Integration Testing.
- ✓ User Acceptance Testing.
- ✓ Output Testing.
- ✓ Validation Testing.

Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program constructions. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly in touch with prospective system users at time of developing and making changes whenever required is done in regard to the following point:

- ✓ Input Screen design
- ✓ Output Screen design

Validation Checking

Validation checks are performed on the following fields:

Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes an error message.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program. The existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test in one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Specifications Testing:

To perform specification testing, the analyst examines the program specifications where in states what program should do and how it should perform under various conditions or combination of conditions and submitted for processing. By examining the results, the analyst can determine whether the program performs according to its specified requirements. This testing does not look into the program to study the code or path it looks at the program as a whole. The assumption here is that, if the program meets the specifications it will not fail.

Output Testing

After performing validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence output format is considered in two ways:

- ✓ Screen
- ✓ Printed format

5.2 SYSTEM IMPLEMENTATION

After proper testing and validation, the question arises whether the system can be implemented or not. Implementation includes all those activities that take place to convert from old system to new one. The new system may be totally new replacing an existing manual or automated system, or it may be a major modification to an existing system. In other case, proper implementation is essential to provide a reliable system to meet organization requirements.

Implementation is the stage of the project where the theoretical design is turned into a working system. At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned a controlled it can cause chaos and confusion.

Implementation includes all those activities that take place to convert from the old system to the new one. The new system may be totally different, replacing an existing manual or automated system or it may be a major modification to an existing system. Proper implementation is essential to provide a reliable system to meet the organization requirements. Successful implementation may not guarantee improvement in the organization using the new system, but improper installation will prevent it.

The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after thorough testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system.

The most crucial stage is achieving a new successful system and giving confidence on the new system for the user that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover.

The more complex the system being implemented, the more involved will be the system analysis and the design effort required just for implementation. The system implementation has three main aspects. They are education and training, system testing and changeover.

The implementation stage involves following tasks:

- ✓ Careful Planning.
- ✓ Investigation of system and constraints.
- ✓ Design of methods to achieve the changeover.
- ✓ Training of the staff in the changeover phase.
- ✓ Evaluation of the changeover method.

The method of implementation and the time scale to be adopted are found out initially. Next the system is tested properly and the same time users are trained in the new procedures.

SYSTEM MAINTENANCE

A system should be created whose design is comprehensive and farsighted enough to serve current and projected user for several years to come. Part of the analyst's expertise should be in projecting what those needs might be in building flexibility and adaptability into the system.

The better the system design, the easier it will be to maintain and the maintenance costs is a major concern, since software maintenance can prove to be very expensive.

Its important to detect software design errors early on; as it is less costly than if errors remain unnoticed until maintenance is necessary. Maintenance is performed most often to improve the existing software rather than to respond to crisis or system failure.

As user requirements changes, software and documentation should be changed as part of the maintenance work. Maintenance is also done to update software in response to the change made in an organization. This work is not as substantial as enhancing the software, but it must be done. The system could fail if the system is not maintained properly.

Software maintenance is of course, far more than “finding mistakes”. We may define maintenance by describing fpor activities that are undertaken to after a program is released for use.

Corrective Maintenance

The first maintenance activity occurs because it is unreasonable to assume that software testing will uncover all latent errors in a large software system. During the use of any large problem, errors will occur and be reported to the developer. The process that includes the diagnosis and correction of one or more errors is called corrective maintenance.

Adaptive Maintenance

The second activity that contributes to a definition of maintenance occurs because of the rapid change that is encountered in every aspect of computing. Therefore, adaptive maintenance- an activity that modifies software to properly interface with a changing environment is both necessary and commonplace.

Perfective Maintenance

The third activity that may be applied to a definition of maintenance occurs when a software package is successful. As the software is used, recommendations for the new capabilities, modifications to existing functions and general enhancements are received from users. To satisfy requests in this category, perfective maintenance is performed. This activity accounts for the majority of all effort expended on the software maintenance.

Preventive Maintenance

The fourth maintenance activity occurs when software is changed to improve future maintainability or reliability, or to provide a better basis for future enhancements. Often called preventive maintenance, this activity is characterized by reverse engineering and re-engineering techniques.

6. CONCLUSION

The system development life cycle comes to a completion with final handover of the system. The system objectives underlined during the start of the system life cycle were all fully met with. The system tries to implement most of the suggestions given by the user during the system study. The others, which are not implemented, due to conflicts were left for future enhancements for the system. All the options were weighed upon the best possible were taken into consideration and implemented.

The system development was a wonderful experience and it has given me a deep insight into real time development. It has also allowed me to uncover the advanced programming concepts of VC++ and C. Maximum justifications are being done to each module in the allotted time span.

7. SCOPE FOR FUTURE DEVELOPMENT

Realtime Interface for Simputer may be enhanced by:

- ✓ The energy meters, if provided with the Serial or USB port the data can be directly downloaded from it.
- ✓ The data from the simputer can be transferred to the PC using some wireless connection technologies like GPRS or IR transfer.

The system if enhanced to meet world-class standards will be used around the globe.

8. BIBLIOGRAPHY

Books:

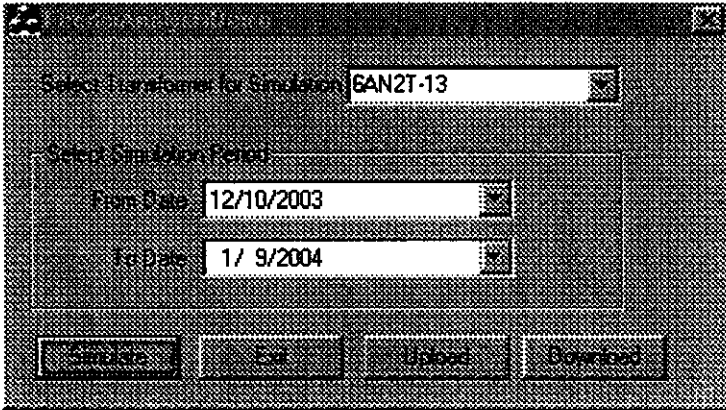
- ✓ “Eric Harlow”, “Developing Linux Application with GTK+ and GDK”, “Techmedia”, 1st Edition, 1999
- ✓ “Andrew S. Tanenbaum”, “Computer Networks”, “Pearson Education”, 4th Edition, First Indian Reprint 2003
- ✓ “W.Richard Stevens”, “Unix Network Programming”, “Pearson Education”, 2nd Edition, Eighth Indian Reprint 2004
- ✓ “Chris H. Pappas & William H. Murray III”, “The Complete Reference Visual C++ 6”, “Tata-McGraw Hill”, Tata-McGraw Hill Edition 1999, Eleventh reprint 2003
- ✓ “Yashavant Kanetkar”, “VC++ Gems”, “BPB Publications”, 1st Edition, 2002

Websites:

- ✓ www.ncoretech.com
- ✓ www.simputer.org

9. Appendix

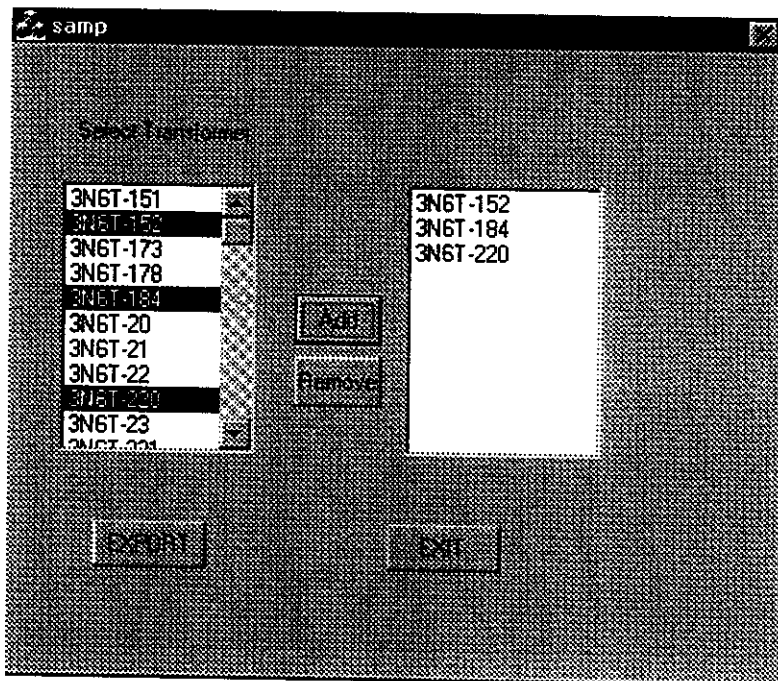
9.1 Screen Shots



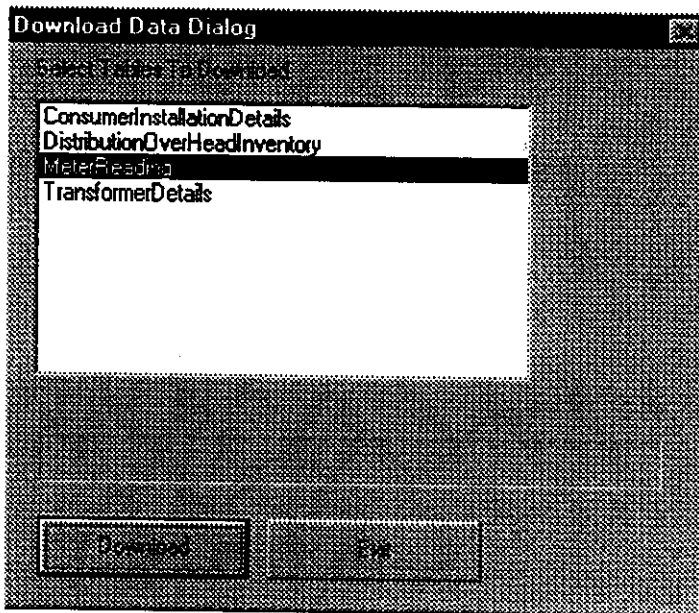
The screenshot shows a software window titled "Select Transformer for Simulation". It contains the following elements:

- A label "Select Transformer for Simulation" followed by a dropdown menu showing "6AN2T-13".
- A label "Select Simulation Period" followed by a rectangular frame containing two date input fields:
 - "From Date" with the value "12/10/2003".
 - "To Date" with the value "1/ 9/2004".
- Four buttons at the bottom: "Simulate", "Exit", "Update", and "Download".

Startup Form



Export data to Simputer from PC



Import data from the Simputer to PC

Consumer Installation

Info Category MeterInfo EnergyMeter

Consumer Installation Details

DTC Ref. No	DTCNO	▼
Division		▼
Sub-Division		▼
FPB/Pole No.		▼
Name		
Address		
Date of visit	2004-8-24	
RR No		
Reading Date	1	▼

Consumer Installation Details (LINUX)

Overhead Inventory

Page1 Page2 Page3 Page4 Page5

Distribution Overhead Inventory

Circle		▼
Division		▼
Sub-Division		▼
Date	2004-8-24	
Location		
DTC Ref.No.	DTCNO	▼
Pole No.		▼
Type of pole	RCC	▼
Span Length	1	▼
Rabbit	1	▼

Distribution OverHead Inventory(LINUX)

Total Connected Load				
Page1	Page2	Page3	Page4	Page5
Distribution Overhead Inventory				
TubeLight (Elec. Choke)	1			
TubeLight (Ord. Choke)	1			
G.I. S. Belt	1			
Table Fan	1			
Ceiling Fan	1			
Pedestal Fan	1			
Exhaust Fan (Ordinary)	1			
Exhaust Fan (Industrial)	1			
Electric Iron	1			
Doal Iron	1			

Total Connected Load (LINUX)